„GoTo“ (GPS-basierter online Tourguide)

Konzeption und Implementierung eines virtuellen Reiseführers auf einer Linux Car-PC Plattform

Diplomarbeit

zur Erlangung des akademischen Grades

Diplom-Informatiker (FH)

vorgelegt am Fachbereich Mathematik, Naturwissenschaften und Informatik der Fachhochschule Giessen-Friedberg, University of Applied Sciences.

Referent: Prof. Dr. W. Schmitt
Korreferent: Prof. Dr. P. Kneisel

In Zusammenarbeit mit der Siemens VDO Automotive AG, Wetzlar.

Betreuer: Dr. Martin Rudolph
Dr. Joachim Tröll

Christian Hainz, im September 2002





Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass

ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegeben Hilfsmittel angefertigt habe,

die aus fremden Quellen direkt oder indirekt übernommenen Gedanken als solche kenntlich gemacht sind.

Dieses Exemplar mit der beurteilten Arbeit übereinstimmt und

diese Arbeit bisher in gleicher oder ähnlicher Form keiner anderen

Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht wurde.

Wetzlar, den 05. September 2002 ______________________________

(Christian Hainz)

Inhaltsverzeichnis

Abkürzungen

AM	Amplituden Modulation
ANSI	American National Standards Institute
ASCII	American Standard Code for Information Interchange
BAB	Bundesautobahn
BSD	Berkeley Software Distribution
CCI	CARiN Communication Interface
CARiN	Car Information and Navigation
DGPS	Differential GPS
DMS	Degrees Minutes Seconds
DVD	Digital Versatile Disc
GMT	Greenwich Mean Time
GoTo	GPS-basierter online Tourguide
GPRS	General Packet Radio Services
GPS	Global Positioning System
GSM	Global System for Mobile communication
HSCSD	High-Speed Circuit-Switched Data
HTML	Hypertext Markup Language
MC5400	Multimedia Center 5400
MIPS	Million Instructions per Second
MMI	Men Machine Interface
MOS	Mean Opinion Score
NMEA	National Marine Electronics Association
PCMCIA	Personal Computer Memory Card International Association
RMC	Recommended Minimum Specific GPS/Transit Data
RTOS	Real-Time Operating System
SGML	Standard Generalized Markup Language
TFT	Thin-film Transistor
TTS	Text to Speech
UHF TV	Ultrahochfrequenz Television
USB	Universal Serial Bus
UTC	Universal Time Coordinated
WGS84	World Geodetic System 1984
XML	Extensible Markup Language

Terminologie

Text to Speech System wird im Wechsel mit **Sprachsynthese System** verwendet. Beide Begriffe bezeichnen die Zusammensetzung von Sprache aus einem Computer gespeicherten Text.

MC5400 ist die kommerzielle Bezeichnung der verwendeten Hardwareplattform, die bei Siemens VDO entwickelt wird.

Touristische Hinweisschilder sind braune Hinweisschilder, zu finden an Bundesautobahnen. Sie weisen auf Sehenswürdigkeiten in Nähe der Autobahn oder einer angrenzenden Stadt hin.

Vorwort

Kurzfassung

Diese Arbeit beschäftigt sich mit der Konzeption und Entwicklung eines virtuellen Reiseführers. Reale Reiseführer, die man als Buch oder Person kennt, werden in Form von Software nachgebildet. Als Hardwareplattform dient ein Linux Car-PC, der in nahezu jedem Fahrzeug eingebaut werden kann. Es handelt sich also nicht um eine Desktop, sondern eher um eine mobile Anwendung.

Ortsgenau werden Reiseinformationen an die Benutzer des virtuellen Reiseführers weitergegeben. Hierbei kann zwischen verschiedenen Funktionen gewählt werden. Im einfachsten Falle werden Informationen zu Sehenswürdigkeiten vorgelesen, die sich zufällig in der Nähe des Fahrzeugs befinden. Besteht eine Verbindung zu einem Navigationssystem, so können die Benutzer sogar auf einer geplanten Reiseroute geführt werden. Unabhängig von der Fahrzeugposition können Informationen, zu den gespeicherten Sehenswürdigkeiten, mittels Browser abgerufen werden. Um den virtuellen Reiseführer aktuell zu halten ist es möglich neue Reisedaten nachträglich zu laden. Für das Vortragen der Reiseinformationen wird ein Sprachsynthese-System integriert.

Im Vordergrund steht nicht die Implementierung eines vollendeten Softwareprodukts. Ziel ist es die einzelnen Teilprobleme aus der Aufgabenbeschreibung zu lösen. Ist für bestimmte Teile eine konkrete Softwarelösung, in der Bearbeitungszeit, nicht möglich so ist zumindest ein Konzept zur Lösung vorgestellt. Zu Demonstrationszwecken ist ein virtueller Reiseführer, mit verringerten Leistungsmerkmalen implementiert.

Abstract

This Diploma Thesis deals with the conception and development of a virtual travel guide. Real travel guides, which one knows as a guidebook or person, are reproduced in software. A Linux Car-PC, that can be built-in any vehicle, is used as hardware platform. Hence, it is rather an automotive than a desktop application.

Travel information is given to the users of the virtual travel guide on exact positions. It is possible to choose between a variety of functions. The easiest way is to read out information about sights that are near by the car, randomly. If a connection to a navigation system exists it is even possible to guide the users on a planned route. Information about sights can be viewed with a browser, independent from the car position. To keep the virtual travel guide up to date it is possible to download supplementary travel data. A speech synthesis system is integrated to declaim the travel information.

It was not the intention to implement a complete software product. The main target is to solve the partial problems from the task description. If a software solution is not feasible for certain parts, during the editing time, at least a concept for a solution is given. For demonstration purposes a virtual travel guide with reduced features is implemented.

Kapitelübersicht

Zu Beginn dieser Arbeit sind im Kapitel eins die verschiedenen Teilaufgaben beschrieben. Das Umfeld der Arbeit wird anhand der Begriffe „Automotive“ und „Embedded Systems“ erklärt. Ausserdem folgt eine ausführlichere Beschreibung der Basistechnologien GPS und Sprachsynthese. Besitzt der Leser eine hinreichend große Vorkenntnis so kann dieser Teil übersprungen werden.

Im zweiten Kapitel findet eine ausgedehnte Betrachtung der Systemkomponenten statt. Einzelne Hard- und Softwareteile, des virtuellen Reiseführers, werden genau analysiert. Im wesentlichen geht es darum, die Schnittstellen der verwendeten Komponenten zu untersuchen. Für diesen Zweck wurden bereits Softwarekomponenten entwickelt, die für die spätere Implementierung als Prototypen dienten.

Der Hauptteil dieser Arbeit ist im Kapitel drei, als Entwicklung des virtuellen Reiseführers, beschrieben. Nach Klärung des Begriffs „Reiseführer“ folgt ein Unterkapitel zum Softwareentwurf. Mit verschiedenen Techniken der Softwareentwicklung werden hier die Voraussetzungen für eine erfolgreiche Implementierung geschaffen. Diese ist im letzten Unterkapitel des Hauptteils, als Lösung der Aufgaben, angegeben. Die anfänglich beschriebenen Teilaufgaben werden hier erneut aufgegriffen und mit Lösungen oder Lösungsvorschlägen versehen. Es kann konkret nachvollzogen werden an welcher Stelle bzw. in welcher Klasse eine Teilaufgabe implementiert wurde.

Abschließend sind Verzeichnisse und der Anhang zu finden. Im ersten Teil des Anhangs sind alle Klassen des virtuellen Reiseführers beschrieben, ohne aber den gesamten Quellkode zu listen. Für die Installation des Sprachsynthese-Systems „Festival“ steht eine Beschreibung in Stichpunkten zur Verfügung. Von besonderem Interesse sollte das ausgewählte Beispiel im letzten Teil des Anhangs sein.

Danksagung

Mein besonderer Dank gebührt den Herren Prof. Dr. Wolfgang Schmitt, Dr. Martin Rudolph und Dr. Joachim Tröll, für die intensive Unterstützung und gute Betreuung. Ausserdem Herrn Prof. Dr. Peter Kneisel, der als Korreferent sofort zur Verfügung stand.

Des weiteren möchte ich Sebastian Wendt, Geschäftsführer der KWest GmbH, und seinen Mitarbeitern danken. Ihre Unterstützung beim Arbeiten mit dem MC5400 und Entwickeln der Graphischen Benutzeroberfläche war eine große Hilfe.

Für das zur Verfügung stellen von Kartenmaterial möchte ich Amtfrau Stefani Jungels, vom Hessischen Landesamt für Strassen- und Verkehrswesen danken.

Nicht unerwähnt sollten auch meine Arbeitskollegen, Kommilitonen, Freunde und Bekannte bleiben, die mir mit ihrem Sachverstand, wertvollen Ratschlägen und Anregungen zur Seite standen.

1 Einleitung

1.1 Aufgabenbeschreibung

Im Rahmen dieser Diplomarbeit soll ein Konzept einer Softwarekomponente für einen virtuellen Reiseführer entwickelt und implementiert werden. Die Hardwareplattform für diese Arbeit ist eine offene, Linux basierte, Multimedia Plattform. Sie entstand aus einer Diplomarbeit [S3] und ist derzeit in der Entwicklung bei Siemens VDO, unter dem Arbeitstitel „MC5400".

Das Ziel sind ortsgenaue (GPS-Genauigkeit) Sprachausgaben, die den Fahrer mit entsprechenden Reiseinfos versorgen. Die Reiseinformationen sind auf dem System als Textdateien mit entsprechenden Positionsmarken vorhanden und können über das Internet nachgeladen werden.

Die Textdateien sollen mit Hilfe eines zu integrierenden frei verfügbaren „Text-To-Speech" Servers in sprachliche Hinweise für den Fahrer umgesetzt werden. Entsprechende Positionsdaten müssen durch einen am Multimedia System angeschlossenen GPS-Empfänger ermittelt und mit den Positionsmarken in den Textpassagen synchronisiert werden.

1.2 Das Umfeld

ODER: WAS BEDEUTEN DIE BEGRIFFE "AUTOMOTIVE" UND "EMBEDDED SYSTEMS"?

Beide Begriffe verkörpern stärkere Einschränkungen und enger gesteckte Randbedingungen als man das aus anderen Bereichen, wie zum Beispiel Telekommunikation, Haushalt, IT-Landschaft, etc. gewohnt ist.

1.2.1 Automotive

Der Begriff Automotive, als Randbedingung, beinhaltet jegliche Problematik, die in einem Automobil auftreten kann. Zu den Problemen zählen vor allem die möglichen Temperatur- und Klimaeinflüsse. So wird beispielsweise von Automobilherstellern ein Temperaturbereich von –30° C bis +40° C gefordert. Innerhalb dieses Bereiches muß ein elektronisches Gerät, im Fahrzeug, immer noch einwandfrei funktionieren. Zusätzlich muß ein solches Gerät mit auftretenden Spannungsspitzen und Vibrationen umgehen können.

Durch vielfältigen Einsatz von Elektronik im Auto können Störungen auftreten, die sich gegenseitig beeinflussen. Um dies zu vermeiden, müssen Geräte im Automotiven Bereich einen gewissen Grad an Störsicherheit und Abschirmung gewährleisten. Hierzu werden Messung bezüglich der Elektromagnetischen Verträglichkeit (EMV) durchgeführt. Vor allem für den Fall des Zusammenwirkens mehrerer Geräte müssen die Spezifikationen der gewählten Verbindungsart unbedingt eingehalten werden.

Man bedenke ausserdem die fatalen Folgen die entstehen können wenn beispielsweise Automatikgetriebe, ABS oder Stabilitätskontrolle versagen. Und das nur weil beim telefonieren mit dem Handy keine Außenantenne benutzt wurde.[1]

Natürlich sind auch die Qualitätsansprüche hinsichtlich eingesetzter Software sehr hoch. Automotive Geräte werden meist in hohen Stückzahlen abgesetzt und das nachträgliche Verbessern von Software ist oft nur schwierig oder gar nicht möglich. Ausserdem gilt es dem Anspruch des Kunden an Sicherheit und Qualität im Auto gerecht zu werden.

Zur weiteren Vertiefung empfehlen sich die Websites der „Society of Automotive Engineers"[2] oder die des Verbandes der Automobilindustrie[3].

1.2.2 Embedded Systems

Die bei VDO entwickelten Geräte sind weitgehend in die Gruppe der „Embedded Systems" einzuordnen. Diese Bezeichnung kommt aus dem Englischen und läßt sich in „eingebettete Systeme" übersetzen.

---

[1] Vgl. [W1] Südwest Rundfunk – Elektronik im Auto

[2] [W2] http://www.sae.org

[3] [W8] http://www.vda.de

Eingebettete Systeme enthalten in der Regel einen oder mehrere Mikroprozessoren, oft auch eine interne Uhr mit Batterie. Das Institute of Electrical and Electronics Engineers (IEEE) definiert „embedded systems“ als „Geräte, die benutzt werden zum Kontrollieren, Überwachen oder Assistieren beim Betrieb von Einrichtungen, Maschinen oder Fabrikanlagen“[4]. Also, im weitesten Sinne auch das Assistieren beim Betrieb eines Kraftfahrzeuges. Hierbei wird häufig Wert auf den Einsatz von Echtzeitsystemen gelegt.

Viele dieser Geräte sind kompakte Computer, die nur ein eingeschränktes Repertoire an Funktionalität besitzen und Beschränkungen bei RAM/ROM Kapazitäten haben. Selten können komplette Befehlssätze in den schnelleren RAM Bereich geladen werden, wodurch die Geschwindigkeit zusätzlich verlangsamt wird. Im Automotive Bereich werden dabei noch hohe Anforderungen hinsichtlich Temperatur, Lebensdauer und Vibration gestellt.

Das Programmieren von Embedded Systems verlangt besondere Sorgfalt und vor allen Dingen überlegte Nutzung der meist knappen Ressourcen, wie Rechenleistung und Speicher was in Kapitel 1.3.2 deutlich wird. Aufgrund hoher Stückzahlen und allgemeinem Preisdruck hat der Entwicklungsaufwand einen geringeren Einfluß, als die Hardware Kosten.

1.3 Basistechnologien

Von den in dieser Arbeit behandelten Kernpunkten GPS, Sprachsynthese, permanente Datenhaltung und Integration des Gesamtsystems sollen die ersten beiden als Basistechnologien betrachtet werden. Durch jahrelange Forschung und finanziell hohe Aufwände stehen heute zwei Technologien zur Verfügung, die für diese Diplomarbeit ganz wesentlich sind. Deshalb wird in den beiden folgenden Kapiteln GPS bzw. Sprachsynthese im Allgemeinen beschrieben.

1.3.1 GPS

GPS steht für Global Positioning System und ist derzeit das einzige funktionierende elektronische System mit dem ein Benutzer seine exakte Position auf der Erdoberfläche bestimmen kann. Dies ist zu jeder Uhrzeit, an fast jedem Ort der Erde und vor allem bei jeder Witterungslage möglich. Gänzlich auszuschließen sind Störungen z.B. durch Abschottung in engen Häuserschluchten oder Reflexionen jedoch nicht.

Es handelt sich um ein Satelliten basiertes System, das gegenwärtig aus 24 Satelliten besteht, welche die Erde in einer Umlaufbahn von ca. 11.000 nautischen Meilen (das sind ca. 12.000 Kilometer) umkreisen. Die Satelliten werden kontinuierlich von Basisstationen überwacht die auf der ganzen Welt verteilt liegen. Von den Satelliten werden Signale ausgesendet, die von jedem empfangen werden können, der einen GPS Empfänger besitzt. Unter Verwendung des Empfängers kann ein Benutzer seine Position sehr präzise bestimmen.
In der Geschichte ist GPS eine der spannendsten und revolutionärsten Entwicklungen der letzten Jahrzehnte. Neue Anwendungen, bei denen GPS verwendet wird, werden ständig entwickelt. Obwohl, GPS zunächst von und für das U.S. Department of Defense entwickelt wurde und nur den Militärs zugänglich war. Durch eine Gesetzesänderung in 1980 schaffte die U.S. Regierung jedoch die Voraussetzungen für eine zivile Nutzung, die bis heute frei von Gebühren ist.



Abbildung 1.1: Erster GPS Satellit GPS Block I

[4] Vgl. [W3] http://www.ieee.org

Bevor aber weiter auf GPS eingegangen wird, erscheint es sinnvoll, die Grundlagen und Notwendigkeiten von Navigation zu verstehen.

1.3.1.1 Grundlagen der Navigation

Seit prähistorischen Zeiten versuchen Menschen einen zuverlässigen Weg zu finden, der ihnen sagt, wo sie sich befinden, sie zu einem Ziel führt und wieder nach Hause bringt. Höhlenmenschen verwendeten möglicherweise Steine und Zweige um ihren Weg zu markieren, wenn sie sich auf der Suche nach Essen befanden. Die frühen Seefahrer orientierten sich an der Küstenlinie um nicht vom Weg abzukommen. Als die ersten Seeleute[5] in offene Ozeane segelten stellten sie fest, das sie sich an den Sternen orientieren konnten. Unglücklicherweise sind die Sterne nur bei Nacht zu sehen und auch nur dann, wenn diese klar genug ist.

Die nächsten großen Entwicklungen bei der Suche nach einer perfekten Methode zur Navigation waren der magnetische Kompass und der Sextant. Die Nadel eines Kompass zeigt immer nach Norden, wodurch es möglich ist, die Bewegungsrichtung zu bestimmen. Ein Sextant verwendet einstellbare Spiegel, um den exakten Winkel zwischen den Sternen, dem Mond und der Sonne über dem Horizont zu messen. Zu Beginn war es aber nur möglich, die geographische Breite (Latitude[6]) mit dem Sextant zu bestimmen. Seefahrer waren aber immer noch nicht in der Lage ihre geographische Länge (Longitude[7]) zu ermitteln. Dies war ein so ernsthaftes Problem, dass die Briten im 17. Jahrhundert einen speziellen „Ausschuß der Longitude“ gründeten, dem viele bekannte Wissenschaftler angehörten. Dieser Ausschuß bot demjenigen 20.000 Pfund[8], der einen Weg fand, die Longitude eines Schiffs innerhalb von 30 nautischen Meilen zu bestimmen.



Abbildung 1.2: Alte Seekarte

Das großzügige Angebot hatte sich gelohnt, als im Jahre 1761 der Möbelschreiner John Harrison eine An Bord Uhr, genannt Chronometer, entwickelte. Harrison’s Chronometer wich pro Tag lediglich eine Sekunde ab – unwahrscheinlich genau für diese Zeit. Über die nächsten zwei Jahrhunderte wurden Chronometer und Sextant gemeinsam verwendet, um Longitude und Latitude Informationen bereit zu stellen.

Im frühen 20. Jahrhundert wurden verschiedene, Rundfunk basierte, Navigationssysteme entwickelt, die ihren weiteren Einsatz während des Zweiten Weltkriegs fanden. Alliierte und feindliche Truppen benutzten bodengestützte Rundfunk-Navigationssystem als fortschrittlichste Technologie.

[5] Das Englische Wort für Seeleute ist Navigators

[6] Lage die nördlich oder südlich zum Äquator gemessen wird

[7] Lage westlich oder östlich zu Greenwich (Stadtteil von London) gemessen wird

[8] Das ist heute, in etwa, vergleichbar mit über einer Million Euro

Einige bodengestützte Rundfunk-Navigationssysteme sind auch heute noch im Einsatz. Ein Nachteil von Radiowellen, die auf dem Boden generiert werden, ist der einzugehende Kompromiss zwischen einem System, das sehr akkurat arbeitet, aber nur eine geringe Fläche abdeckt oder einem System, das weniger genau ist, dafür aber eine weite Fläche bedienen kann. Hochfrequente Radiowellen (wie UHF TV) liefern eine sehr genaue Position können aber nur in einem begrenzten Raum empfangen werden. Niederfrequente Radiowellen (wie AM Radio) hingegen können einen sehr großen Bereich abdecken sind aber keine gute Hilfe beim Feststellen der Position.

Deshalb entschieden Wissenschaftler, dass der einzige Weg, die gesamte Welt abzudecken, die Platzierung von Hochfrequenz Funksendern im Weltall ist. Ein Funksender hoch über der Erde, der hochfrequente Radiowellen mit einem speziell codierten Signal aussendet, kann einen sehr großen Bereich abdecken und dabei die Probleme von Störgeräuschen auf dem Boden überwinden. Dies ist eines der Hauptprinzipien von GPS.

1.3.1.2 Elemente von GPS

GPS besteht aus drei Teilen: Dem Space Segment, dem User Segment und dem Control Segment [W7]. Das Space Segment besteht aus einem Netzwerk von Satelliten, wovon sich jeder in einer eigenen Erdumlaufbahn von ca. 12.000 Kilometern befindet. Beim User Segment handelt es sich um die erwähnten GPS Empfänger die zum Beispiel in der Hand gehalten oder im Auto montiert werden können. Fünf Basisstation, die auf der Welt verteilt sind, bilden das Control Segment und sorgen für die einwandfreie Funktion der Satelliten.
Für ein besseres Verständnis sind in der nachfolgenden Abbildung die verschiedenen Teile des GPS Systems dargestellt.

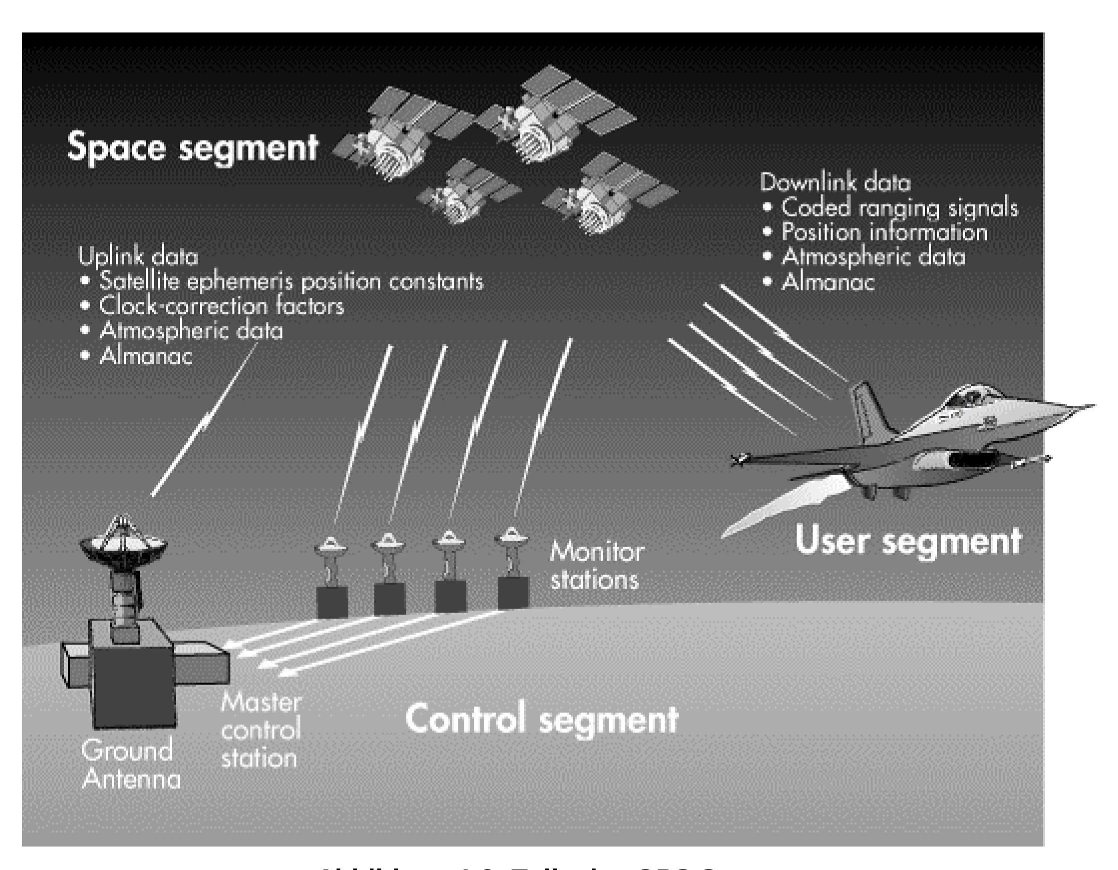


Abbildung 1.3: Teile des GPS Systems

Jeder der GPS Satelliten benötigt 12 Stunden zum Umkreisen der Erde. Alle sind mit einer akkuraten Uhr ausgestattet, um die Übertragungen mit präzisen Zeitangaben zu koppeln. Die Bodeneinheit empfängt das Signal, das mit Lichtgeschwindigkeit übertragen wird. Aber, sogar bei dieser Geschwindigkeit benötigt das Signal eine messbare Zeit um den Empfänger zu erreichen. Der zeitliche Unterschied zwischen Senden und Empfangen, multipliziert mit der Lichtgeschwindigkeit, ergibt den Abstand zum Satelliten. Um die genaue Latitude, Longitude und Altitude[9] zu bestimmen berechnet der Empfänger die Zeitunterschiede zu vier verschiedenen Satelliten.

Das GPS System kann die genaue Position an jeder Stelle der Erde bis auf 100 Meter genau anzeigen. Eine größere Genauigkeit, für gewöhnlich unter einem Meter, kann mit Korrekturrechnungen bezüglich bekannter Punkte erzielt werden. Diese Methode nennt man auch Differential GPS (DGPS).

1.3.1.3 Arbeitsweise von GPS

Das Grundprinzip hinter GPS ist die Berechnung der Distanz zwischen Satellit und Empfänger. Ausserdem teilen die Satelliten mit, wo sie sich in der Umlaufbahn der Erde befinden. Wenn man die exakte Distanz zu einem Satelliten im Weltraum kennt, kann man sich vorstellen, auf der Oberfläche einer fiktiven Kugel mit einem Radius gleich der Distanz zum Satellit zu stehen. Mit dem Wissen einer weiteren Distanz zu einem zweiten Satelliten steht man auf dem Schnittkreis zweier Kugeln. Nimmt man eine dritte Messung hinzu so gibt es nur noch zwei Punkte, auf der Erdoberfläche, an denen man sich befinden kann. Für gewöhnlich ist einer dieser Punkte unmöglich und GPS Empfänger haben mathematische Methoden, um die unmögliche Position auszuschließen.

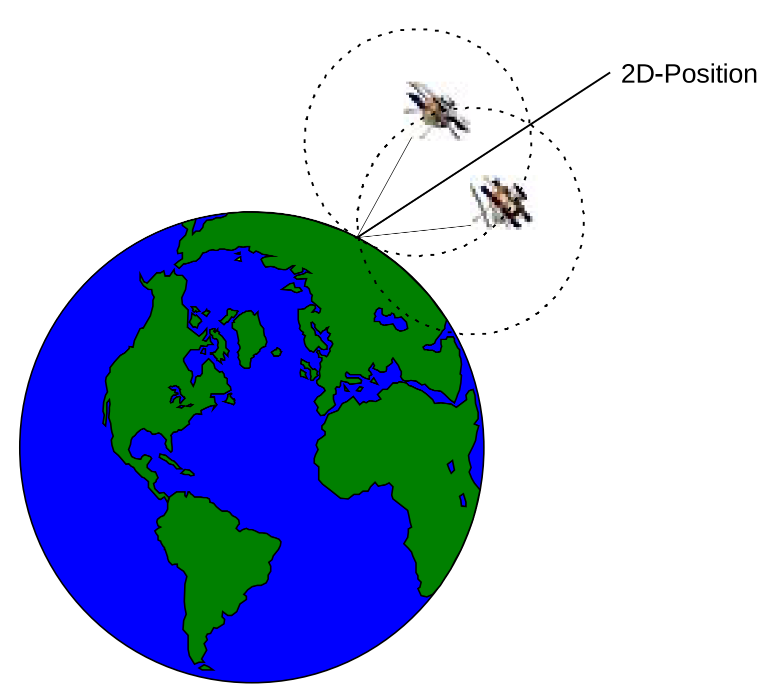


Abbildung 1.4: Feststellen der 2D-Position

1.3.1.4 Verwendetes Koordinaten-System

GPS basiert auf einem Koordinaten-System, das als „World Geodetic System“ von 1984 (WGS84) bekannt ist. Es ist ähnlich zu den Längen- und Breitengraden, die man als Linien auf Landkarten oder Globen sehen kann.

Typischerweise werden Koordinaten in Winkel, Minuten, Sekunden und eventuell Zehntelsekunden angegeben. Die Umrechnung in einen Winkel, als Gleitzahl, ist leicht möglich. Eine ausführliche Beschreibung des Koordinaten-Systems ist zu finden unter [W13].

---

[9] Höhe über dem Meer

1.3.2 Sprachsynthese

Sprache ist die meist verwendete Form der Kommunikation. Sprachsynthese, das automatische Generieren von Sprache, ist seit Jahrzehnten in der Entwicklung. Die jüngsten Fortschritte in der Sprachsynthese haben Synthetisierer mit hoher Verständlichkeit hervorgebracht. Klangqualität und Natürlichkeit bleiben jedoch weiterhin die größten Probleme. Dies gilt insbesondere für die Embedded Systems bei denen, wie eingangs beschrieben, Rechenleistung und Speicher nur eingeschränkt verfügbar sind. Davon einmal abgesehen, haben gegenwärtige Produkte eine angemessene Qualität für verschiedene Anwendungen, wie zum Beispiel Multimedia und Telekommunikation erreicht. Zusammen mit audiovisuellen Informationen oder Gesichtsanimationen ist es möglich, die Verständlichkeit noch bedeutend zu steigern.

Der Ablauf der „Text-To-Speech“ (TTS) Synthese besteht aus zwei Hauptphasen. Die erste Phase ist die Textanalyse, wo der Eingabetext in Lautschrift, oder eine andere Sprachdarstellung überführt wird. In der zweiten Phase findet die Generierung von Sprache statt, wobei eine akustische Ausgabe aus den Lautschrift Informationen produziert wird.
Diese zwei Phasen werden üblicherweise als „high- und low-level“ Synthese bezeichnet [S2]. Ein vereinfachtes Modell dieses Vorgangs ist in Abbildung 1.5 dargestellt. Der Eingabetext könnte beispielsweise von einem Word Prozessor[10] stammen, standard ASCII[11] Text einer E-Mail sein, eine Kurznachricht eines Mobiltelefons oder ein Zeitungsartikel, der gescannt wurde. Der Zeichenstrom wird dann vorverarbeitet und analysiert, so dass eine Repräsentation in Lautschrift vorliegt. Üblicherweise ist das eine Kette von Phonemen[12] mit zusätzlichen Informationen für korrekten Tonfall, Dauer und Betonung. Sprachausgaben werden letztendlich, unter Zuhilfenahme der high-level Informationen, mit dem low-level Synthetisierer generiert.

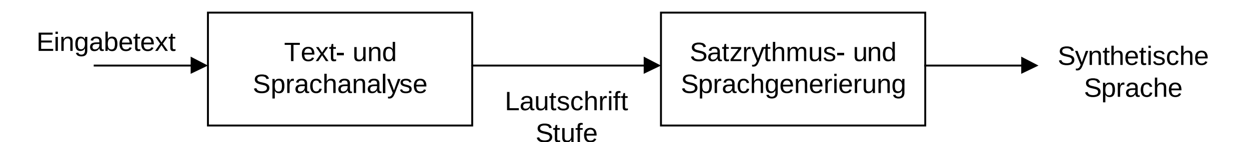


Abbildung 1.5: Einfacher Text-To-Speech Synthese Vorgang

Der einfachste Weg synthetische Sprache zu produzieren, ist das Abspielen von möglichst langen vorher aufgenommenen Stücken natürlicher Sprache, wie Worte oder komplette Sätze. Diese Methode des Zusammenhängens bietet hohe Qualität und Natürlichkeit, besitzt aber ein begrenztes Vokabular und funktioniert für gewöhnlich nur für eine Sprache. Sie ist sehr geeignet für Ansage- und Informationssysteme und wird derzeit auch in Navigationssystemen eingesetzt.

Für den Automotive Bereich wird es aber immer erforderlich sein möglichst viele Sprachen gleichzeitig zu unterstützen. Es wird auch sehr schnell klar, dass es nicht möglich ist, eine Datenbank mit allen Worten und gebräuchlichen Namen dieser Welt zu schaffen. Möglicherweise ist die Bezeichnung Sprachsynthese für diese Methode sogar unzutreffend, da lediglich Aufzeichnungen wieder gegeben werden.
Also, für uneingeschränkte Sprachsynthese (Text-To-Speech) müssen kürzere Stücke von Sprachsignalen wie Silben oder Phonemen verarbeitet werden.

Eine andere weit verbreitete Methode, um synthetische Sprache zu produzieren ist die Formantsynthese[13]. Sie basiert auf dem Source-filter Model der Sprachproduktion. Dies ist in Abbildung 1.6 skizziert.

---

[10] Microsoft © Word © wäre zum Beispiel ein solcher

[11] ASCII (American Standard Code for Information Interchange) ist das meist verwendete Format für Textdateien in Computern. In einer ASCII-Datei wird jeder Buchstabe, jede Zahl oder jedes Sonderzeichen als Binäre 7-bit Zahl dargestellt. 128 mögliche Zeichen sind dabei wohl definiert. ASCII wurde entwickelt vom American National Standards Institute (ANSI). Tabelle zu ASCII wurde entnommen aus [B7] S. 251.

[12] Ein Phonem ist die kleinste bedeutungsunterscheidende lautliche Einheit einer Sprache [W5], [W6]

[13] Teil des Frequenzspektrums der Sprache [W6]. Die Überlagerung von drei niedrigen Formants (F1, F2 und F3) haben einen hohen Anteil beim Produzieren von unterscheidungskräftigen Vokalen und vielen Konsonanten

Diese Methode wird zeitweise auch als endständige Analogie bezeichnet, weil sie nur die Tonquelle und die Formantfrequenz modelliert, aber keine physikalischen Eigenschaften der Stimmwege.
Das Signal zur Sprachanregung kann entweder stimmhaft zusammen mit der Grundfrequenz (F0) sein oder stimmlose Geräusche. Eine gemischte Anregung durch beide kann benutzt werden, um stimmhafte Konsonanten zu erzeugen. Das anregende Signal wird dann verstärkt und mit einem Stimmwegsfilter gefiltert. Der Stimmwegsfilter wird aus Resonatoren, ähnlich zu denen natürlicher Sprache, aufgebaut.

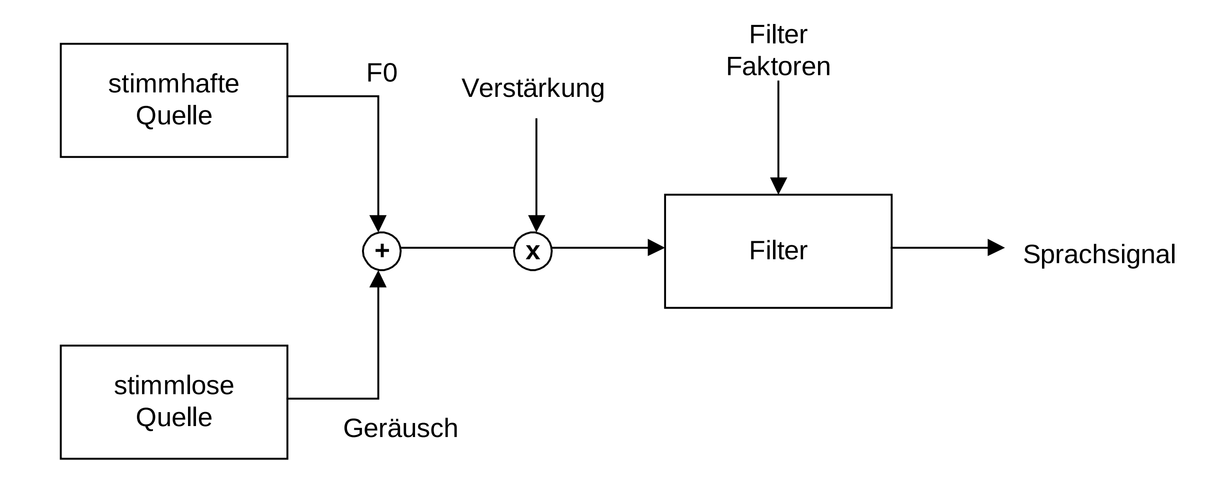


Abbildung 1.6: Source-filter Modell der Sprache

Die theoretisch beste Methode zur Sprachgenerierung erreicht man durch direkte Modellierung des menschlichen Sprachsystems. Diese Methode, als Synthese der Artikulation [S2] bekannt, schließt typischerweise die Modellierung der menschlichen Artikulatoren und der Stimmbänder ein. Die Synthese der Artikulation verspricht eine qualitativ hoch synthetisierte Sprache, aber aufgrund der hohen Komplexität wurde das Potential bisher nicht voll ausgeschöpft.

Für alle Synthese Methoden gilt, dass jede ihre eigenen Vorteile und Probleme hat und dass es schwer ist zu sagen, welche Methode die beste ist. Allerdings lassen sich Bewertungskriterien aufstellen um Sprachsynthese Systeme miteinander zu vergleichen[14].
Oft werden technische Eckdaten wie Speicherbedarf, der sogenannte „Memory Footprint“ und die benötigte Rechenleistung in MIPS[15] zuerst verglichen. Danach folgt ein Vergleich verschiedener Systeme nach der „Mean Opinion Score“ (MOS). Hierbei wird einem ausgewählten Hörerkreis eine Sprachprobe jedes zu bewertenden Systems vorgespielt. Die Hörer entscheiden nach ihrem persönlichen Empfinden über Aussprache, Betonung, Geschwindigkeit, Deutlichkeit, Verständlichkeit und Natürlichkeit. Ergebnisse werden auf einer Skala von eins bis fünf eingetragen, wobei jeder Wert unter drei als durchgefallen gilt und die Höchstmarke von fünf der üblichen HiFi-Qualität entspricht. Weitere Kriterien, wie verfügbare Sprachen und Kosten eines Systems können natürlich auch in einem Benchmark verglichen werden.

Die Auswahl und Bewertung eines Sprachsynthese Systems war nicht Gegenstand dieser Arbeit. Vielmehr wurde das System benutzt welches zum Zeitpunkt dieser Arbeit frei verfügbar und unter angemessenem Aufwand integrierbar war. Eine Betrachtung der Schnittstellen zum benutzten System ist im Hauptteil, Kapitel 2.4, zu finden.

---

[14] Ein solcher Vergleich wird häufig als „Benchmark” bezeichnet

[15] Die Anzahl der Million Instructions per Second (MIPS) wird allgemein verwendet, um die Rechenleistung von Computern zu vergleichen

2 Betrachtung der Systemkomponenten

2.1 Gesamtübersicht

Im folgenden Abschnitt soll zunächst eine grobe Übersicht einzelner Systemkomponenten gegeben werden. Die in der Einleitung angesprochene Hardwareplattform, Multi Media Centre MC5400, bildet dabei den Kern des Gesamtsystems. Zur Veranschaulichung dient die Abbildung 2.1.

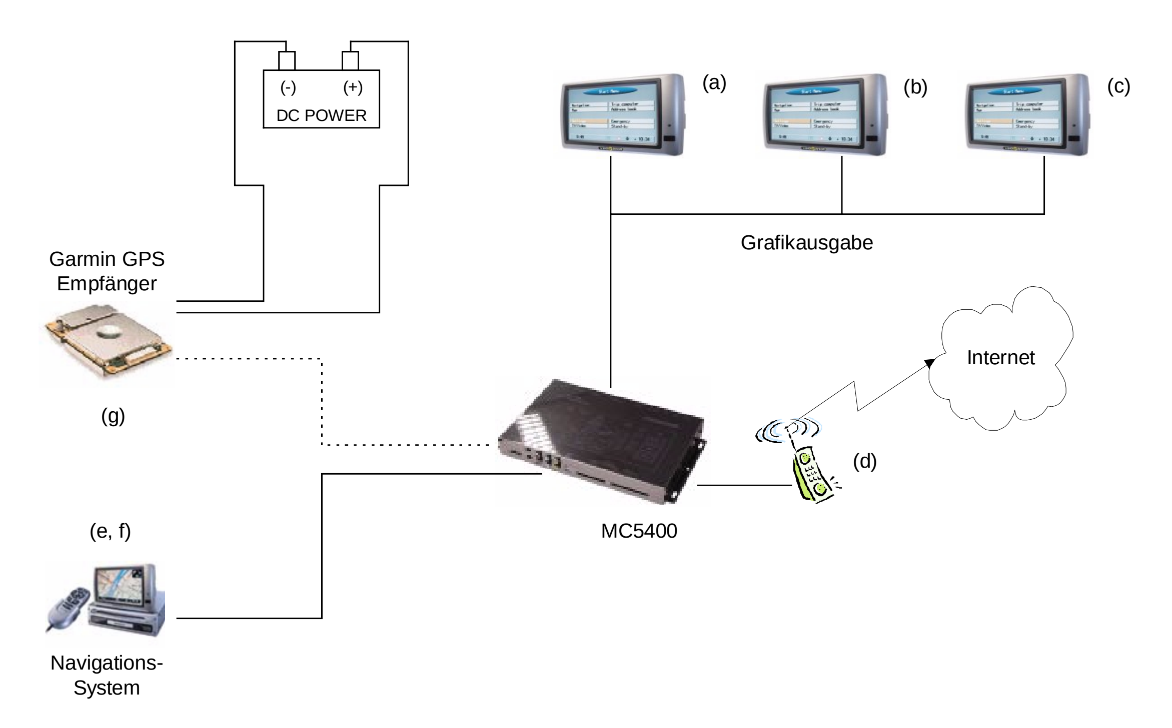


Abbildung 2.1: Systemübersicht

Um das MC5400 (d) sind weitere Hardwarekomponenten gruppiert. Zur Grafikausgabe dienen bis zu drei TFT Monitore (a, b, c), die im 16:9 Format Farbbilder mit hoher Qualität wiedergeben und außerdem einen eingebauten Lautsprecher besitzen. Auf die Monitore wird im folgenden nicht weiter eingegangen. Für den interessierten Leser empfiehlt sich die technische Dokumentation der Firma Sharp [S1].

Das Navigationssystem (f), inklusive Fernbedienung (e), ist ein Produkt der Firma Siemens VDO und kann mit einer speziellen Software betrieben werden. Diese ermöglicht die Kommunikation mit dem Navigationssystem über eine serielle Schnittstelle, nach RS-232. Im einfachsten Fall wird das Navigationssystem als Lieferant für aktuelle und sehr genaue Positionsdaten verwendet.

Als Alternative für die Positionsbestimmung kann ein externer, handelsüblicher, GPS Empfänger (g) verwendet werden. Keine der beiden GPS Quellen wird einen Anspruch auf 100%ige Genauigkeit erheben können. Wertet man die Positionsdaten beider Quellen an einem bestimmten Ort aus, so wird man eine Abweichung feststellen. Eine genauere Betrachtung dieser Problematik wird in einem folgenden Kapitel gegeben.

Beide Arten der Positionsbestimmung haben eine kommerzielle Berechtigung. Durch Verwendung des Navigationssystems wird der integrale Systemaspekt gefördert, da Navigationssystem und Multimedia Center vom gleichen Hersteller stammen. Allerdings ist die Anschaffung des Navigationssystems ein erheblicher Kostenfaktor. Durch Verwendung des wesentlich billigeren GPS Empfängers kann der virtuelle Reiseführer, mit verringerten Leistungsmerkmalen, auch ohne angeschlossenes Navigationssystem betrieben werden.

Außer den offensichtlichen Hardwarekomponenten sind noch verschiedene Teile der Software hervorzuheben. Das Sprachsynthese System (Text to Speech System) dient zur akustischen Aufbereitung von Reiseinformationen.

Die auszugebende Reiseinformation erhält das Sprachsynthese System aus einer Sammlung permanent gespeicherter Daten. Übergeben wird eine Information als ASCII codierter Text. Der Begriff „Datenbank“ wird in folgenden Ausführungen zur Veranschaulichung verwendet.

Für ein zukünftiges System soll es außerdem möglich sein, die Datenbank per Internet zu aktualisieren. Hierzu kann eine Internet-Konnektivität über mobile Kommunikation (d) mittels GSM, GPRS oder HSCSD genutzt werden. Dies ist jedoch nicht Inhalt dieser Arbeit.

Ein mit dem MC5400 stark gekoppelter Teil der Software ist die graphische Benutzeroberfläche. Die im Folgenden als MMI (Men Machine Interface) bezeichnete Schnittstelle zum Benutzer, wurde komplett an das bestehende System angepasst. Dabei wird eine nahtlose Einbindung in das bisherige Erscheinungsbild gewährleistet. Alle Funktionen des virtuellen Reiseführers können interaktiv über die MMI gesteuert werden.

2.2 Embedded System MC5400

2.2.1 Beschreibung der Hardwareplattform

Das Multi Media Centre MC5400 entstand aus einer Diplomarbeit, die im September 2000 abgeschlossen wurde. Der damalige Titel lautete „Evaluation eines Embedded PC-Systems als Multimediaplattform im Kraftfahrzeug“ [S3]. Seit dieser Zeit befindet sich die Hard- und Software in der Entwicklung zur Produktreife und steht kurz vor der Markteinführung. Ziel ist die einheitliche Bedienung von Videoquellen im Auto. Alle Geräte sollen mit einer Fernbedienung steuerbar sein. Ausserdem stehen noch viele Eigenfunktionen wie Computerspiele, Fax senden/empfangen, Internet browsen, E-Mail Verkehr und das Abspielen von MP3[16], zur Verfügung. Es dient als Plattform für weitere Funktionalitäten wie zum Beispiel der aus dieser Arbeit resultierenden Reiseführung.

Als Hardware Basis wird der STPC-Mikrocontroller der Firma ST Microelectronics verwendet. Dieser Prozessor ist pin- und funktionskompatibel zu Intel’s 80486DX. Es steht Hauptspeicher von 32MB RAM zur Verfügung und eine große Anzahl an Schnittstellen, von denen nur einige erwähnt werden sollen. Auf einem Standard Board befinden sich u.a. 2x serielle Schnittstelle (RS-232 Kompatibel), 3x Line Out, Aux und DVD in (jeweils Audio L, R & Video), 2x USB, 3x Monitor, Rückfahrtkamera, TV-Empfänger, Compact Flash Card Reader und PCMCIA Slot.

Für diese Arbeit wird mindestens eine der beiden seriellen Schnittstellen zum Anschluß einer GPS-Quelle genutzt. Die zweite serielle Schnittstelle kann zum Verbinden mit einem PC verwendet werden, so dass über ein Terminalprogramm das Zugreifen zur Laufzeit möglich ist. Mindestens ein Monitor sollte angeschlossen sein, um die Graphische Benutzeroberfläche anzeigen zu können. Der Compact Flash Card Reader nimmt eine Flash-Karte mit der Betriebssoftware auf. Weitere Software befindet sich auf einer zweiten Flash-Karte die über einen Adapter im PCMCIA Slot betrieben wird. Zur verbesserten akustischen Ausgabe kann das Audio Signal über Line Out an einen entsprechenden Verstärker geführt werden.

2.2.2 Software-Architektur

Als Betriebssystem wird Linux[17], basierend auf dem Kernel der Version 2.4.3, verwendet. Dieser wurde um einige Funktionen, wie zum Beispiel die Infrarotcode-Dekodierung, erweitert. Da eine Installation von Standard Linux-Distributionen größere Mengen an Speicherplatz erfordert muß ein Minimal-Linux installiert und konfiguriert werden. Dieses Minimal-Linux enthält nur die notwendigsten Systemkomponenten, um mit einem Minimum an Flash-Speicher auszukommen. Daher wird es auch als Embedded Linux bezeichnet. Ausserdem besitzt es einen Echtzeitkern, so dass man von einem RTOS (Real-Time Operating System) sprechen kann. Bei Bedarf können sogar harte Echtzeit-Anforderungen erfüllt werden.

Für die Grafikausgabe wird ein Xfree86-SVGA-Server [W10] in Version 3.3.6 verwendet. Die Grafikbibliothek, welche maßgeblich bei der Gestaltung der MMI verwendet wird, ist Qt 2.3.1[18]. Ausserdem wird die Glibc 2.2.2 verwendet. Als Shell dient die „Busybox“ [W12], die zusätzlich noch einige weitere Befehle, wie zum Beispiel Filesystem-Check, enthält.

Busybox ist besonders geeignet für Embedded Linux, da es für eingeschränkte Ressourcen optimiert wurde. Befehle und Features lassen sich während der Kompilierung ein- oder ausschalten.

---

[16] MP3 (MPEG-1 Audio Layer 3) ist eine Standard Technologie und ein Format für die Komprimierung von Musiksequenzen. Ohne erheblich an Qualität zum original zu verlieren wird eine sehr kleine Datei erzeugt.

[17] Hauptsächlich von Linus Torvald entwickeltes UNIX System für PCs, welches als hinreichend bekannt vorausgesetzt wird. Für Dokumentationen siehe u.a. [W9] und [B6].

[18] Von Trolltech entwickeltes cross-platform C++ GUI toolkit, siehe [W11]

Das Linux-System selber befindet sich auf einer Compact Flash Card, die in den CF Card Reader eingeschoben wird. Ohne eingeschobene Flash Card ist das System nicht lauffähig. Standardmäßig werden 32MB bzw. 64 MB Karten verwendet. Über einen Flash Card Adapter kann eine weitere Karte im PCMCIA Slot „gemountet“ werden. Die Flash Card für den Systembetrieb hat die in Abbildung 2.2 gezeigte Struktur.

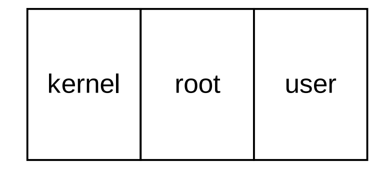


Abbildung 2.2: Partitionen auf der primären Flash Card

Auf der mit „kernel“ bezeichneten Partition befindet sich der Linux Kern. Zum Schutz vor unerlaubten Schreibbefehlen wird diese Partition im Modus „Read only“ gestartet. Alle lauffähigen Programme, der X-Server, Bibliotheken und weiteres befinden sich auf der Partition „root“. Auch diese wird zum Programmschutz nur „Read only“ gestartet. Die Software kann also im laufenden Betrieb nicht verändert werden. Um aber das Abspeichern von Benutzereinstellungen und Fehlermeldungen zu ermöglichen wird die Partition „user“ im Modus „Read and write“ gestartet. Hier können zusätzlich noch spezifische Benutzerdateien, wie z.B. MP3‘s abgelegt werden.

2.3 GPS Quellen

2.3.1 Navigationssystem

In verschiedenen Baureihen produziert Siemens VDO Navigationssysteme, für den Einsatz in Kraftfahrzeugen. In der Regel verfügen alle Systeme über ein CD-ROM Laufwerk, über das kartographische Informationen eingelesen werden. Zur Bestimmung der Fahrzeugposition wird eine GPS-Antenne verwendet. Um hierbei Genauigkeit und Zuverlässigkeit zu erhöhen kommen jedoch weitere Methoden zur Anwendung.

Eine von der GPS-Antenne erhaltene Position wird mit dem digitalen Kartenmaterial verglichen und falls nötig in Richtung der nächstgelegenen Straße korrigiert. Hierbei wird im Allgemeinen davon ausgegangen das sich ein Kraftfahrzeug auf und nicht neben der Straße befindet. Diese Methode wird als Koppelnavigation bezeichnet. Ausserdem wird zur Kurvenerkennung ein Gyroskop-Modul[19] benutzt und das Tachometersignal des Autos wird ausgewertet.

Die wesentliche Schnittstelle zum Navigationssystem bildet das sogenannte „CARiN Communication Interface“ (CCI). Hierbei handelt es sich um ein proprietäres Protokoll, das auf der seriellen Schnittstelle des Navigationssystems realisiert ist. Nach Initialisierung ist es einem externen Gerät, zum Beispiel einem PC oder dem MC5400 möglich, vordefinierte Telegramme mit dem Navigationssystem auszutauschen.

Grundlegende Formate der Telegramme sind in [F3] beschrieben. Implementierung konkreter Telegramme und eine entsprechende Übersicht ist in [F4] zu finden. Für diese Arbeit werden Hauptsächlich die Telegramme verwendet, die notwendig sind um Positionskoordinaten abzufragen, oder Zielkoordinaten zu setzen.

---

[19] Elektromagnetischer Kreiselkompass zur allgemeinen Bestimmung der Bewegungsrichtung. Hierbei wird die Drehung des Fahrzeugs ermittelt und mit dem Kurvenradius auf dem Kartenmaterial verglichen.

2.3.2 GPS Mouse

Die „GPS Mouse“ ist ein kompletter GPS Empfänger, mit eingebauter Antenne, der z.B. von der Firma Garmin hergestellt und vertrieben wird. Basierend auf bewährter Technologie vorheriger Garmin GPS Empfänger, arbeitet die GPS Mouse mit bis zu 12 Satelliten gleichzeitig. Die benötigte Zeit bis zur ersten Positionsbestimmung wird als kurz bezeichnet. Aktualisieren von Positionsdaten findet einmal pro Sekunde statt. Die physikalische Übermittlung von Daten wird über eine serielle Schnittstelle (RS-232 kompatibel) abgewickelt. Alle relevanten Details zur GPS Mouse sind in Garmin's technischer Spezifikation [S4] beschrieben

Das Schnittstellen-Protokoll zur GPS Mouse basiert auf der Interface Spezifikation der „National Marine Electronics Association NMEA 0183 ASCII“[20] und den „Recommended Standards For Differential Navstar GPS Service, Version 2.0, RTCM Special Committee No. 104“[21] der Radio Technical Commission for Maritime Services (RTCM). Zusätzlich zur Übertragung von Navigationsdaten nach NMEA 0183 werden weitere Informationen nach einer proprietären Konvention von Garmin übermittelt.

Übermittelt werden Daten in Form von kommaseparierten ASCII Sätzen. Eine bidirektionale Kommunikation ist möglich, wobei die GPS Mouse kontinuierlich, einmal pro Sekunde, eine Anzahl von Sätzen überträgt. Welche Sätze übertragen werden kann der Benutzer einstellen. Jeder Satz ist durch eine eindeutige Buchstabenfolge gekennzeichnet. Als Beispiel soll der Satz **$GPRMC**, in Abbildung 2.3, betrachtet werden.

Das Dollar Zeichen markiert den Beginn eines neuen Satzes. GPRMC steht hier für Garmin Proprietär (GP) und Recommended Minimum Specific GPS/Transit Data (RMC). In dem Wert *hh wird eine 1 Byte lange Checksumme mit übertragen. Die ASCII Zeichen für Wagenrücklauf (CR, 0x0D) und Zeilenvorschub (LF, 0x0A) schließen einen Satz ab. Dieser Aufbau ist im Prinzip für alle Sätze ähnlich. In den Feldern <1> bis <11> werden hier die Werte für Positionsdaten untergebracht, die im einzelnen in [S4] nachgelesen werden können.

```
$GPRMC,<1>, <2>,<3>, <4>,<5>,<6>,<7>,<8>,<9>,<10>,<11>*hh<CR><LF>
```

Abbildung 2.3: Aufbau des Satzes $GPRMC

2.3.3 Abweichungen bei der Positionsbestimmung

Während dem abwechselnden Arbeiten mit Navigationssystem und GPS Mouse konnten subjektive Abweichungen bei der Positionsbestimmung festgestellt werden. Zunächst wurde vermutet, dass die erfahrenen Abweichungen durch eine Ungenauigkeit der GPS Mouse zustande kommen. Dies könnte passieren wenn nur wenig Satelliten in Sichtweite sind. Im Navigationssystem dagegen spielt die Anzahl der Satelliten in Sichtweite eine untergeordnete Rolle, da wie oben erwähnt zusätzliche Quellen mit einbezogen werden.

Um ein genaueres Bild des Verhaltens zu bekommen, wurde im Fahrzeug eine Versuchsmessung durchgeführt. Bei dem Versuch wurden an zehn verschiedenen Orten Positionsdaten ermittelt. Zunächst per GPS Mouse und nach kurzer Verzögerung mit dem Navigationssystem. Dieser Versuch lieferte die in Tabelle 2.1 abgebildeten Messergebnisse.

---

[20] NMEA 0183, Version 2.0 ist erhältlich bei NMEA, P.O. Box 50040, Mobile, AL, 36605, U.S.A.

[21] Erhältlich bei RTCM, P.O. Box 19087, Washington, D.C., 20036, U.S.A.

#	Ort	Position mit Navigation	# Sat.[22]	Position mit GPS Mouse	# Sat.	Distanz[23] in m
1	VDO vor dem Gebäude A	50° 34‘ 0.9“ N 8° 30‘ 22.8“ E	7	50° 34‘ 5.106“ N 8° 30‘ 20.598“ E	8	137.07
2	A45 Abfahrt Wetzlar Süd, Parkplatz	50° 33‘ 22.3“ N 8° 33‘ 48.3“ E	6	50° 33‘ 23.4“ N 8° 33‘ 47.19“ E	8	40.41
3	A45 Abfahrt Giessen-Lützellinden	50° 31‘ 35.1“ N 8° 35‘ 53.7“ E	7	50° 31‘ 35.112“ N 8° 35‘ 51.582“ E	8	41.64
4	A45 Parkplatz	50° 30‘ 32.9“ N 8° 40‘ 42.4“ E	6	50° 30‘ 31.284“ N 8° 40‘ 40.188“ E	8	66.25
5	B488 Eberstadt	50° 29‘ 4.9“ N 8° 45‘ 57.6“ E	6	50° 29‘ 4.614“ N 8° 45‘ 56.694“ E	8	19.9
6	B488 Parkplatz Kloster Arnsburg, im Wald	50° 29‘ 46.1“ N 8° 47‘ 18.8“ E	6	50° 29‘ 46.59“ N 8° 47‘ 18.264“ E	9	18.46
7	B488 Ortseingang Lich	50° 31‘ 2.1“ N 8° 48‘ 31.7“ E	6	50° 31‘ 2.574“ N 8° 48‘ 31.596“ E	9	14.8
8	Ortseingang Fernwald-Steinbach	50° 32‘ 55.0“ N 8° 46‘ 13.9“ E	6	50° 32‘ 54.888“ N 8° 46‘ 14.382“ E	8	10.09
9	Giessen, Industriegebiet Europaviertel	50° 34‘ 12.3“ N 8° 43‘ 36.7“ E	7	50° 34‘ 13.422“ N 8° 43‘ 37.314“ E	8	36.73
10	B49 P+R Parkplatz Lahnau	50° 34‘ 10.6“ N 8° 33‘ 51.7“ E	6	50° 34‘ 9.78“ N 8° 33‘ 51.588“ E	8	25.45
11	VDO vor dem Gebäude A	50° 34‘ 4.5“ N 8° 30‘ 20.9“ E	6	50° 34‘ 3.84“ N 8° 30‘ 20.322“ E	7	23.35

Tabelle 2.1: Messergebnisse zur Abweichung von GPS-Quellen

Da keine der beiden Methoden als 100% genau angesehen werden kann ist es auch nicht möglich zu sagen, welche Messung die Abweichende ist. Aber mit einem Hintergrundwissen über GPS und Navigationssysteme lässt sich grob eine Erklärung abgeben.

Bei gleichbleibender Anzahl an verfügbaren Satelliten arbeitet die GPS Mouse, an jedem Ort der Welt, mit gleicher Ungenauigkeit[24]. Durch Bewegung über Land verändern sich die Randbedingungen für die GPS Mouse im Prinzip nicht.
Das Navigationssystem hingegen bezieht, außer dem Satellitensignal, noch weitere Faktoren mit ein, die sehr wohl durch eine Fahrbewegung beeinflußt werden. Die Positionskoordinaten des Navigationssystems werden bei vorhandenem Kartenmaterial immer mit diesem abgeglichen. Das bedeutet in erster Konsequenz, dass die Ungenauigkeit auf langen geraden Stücken zunimmt, da weniger Knotenpunkte durchfahren werden. Werden Gebiete durchfahren die nicht in digitaler Form kartographiert sind nimmt die Ungenauigkeit sogar noch mehr zu.
Die Abweichungen in Messung eins und elf sind interessant, da sie an der selben Stelle durchgeführt wurden, aber stark veränderte Abweichungen ergeben. Während die Werte der GPS Mouse relativ stabil geblieben sind, haben sich bei dem Navigationssystem andere eingestellt. Erklärt werden kann dies durch die entsprechende Vorgeschichte. In Messung eins wurde das Navigationssystem im Stand gestartet, ohne das ein digitalisiertes Gebiet durchfahren wurde. Der Messung elf ging jedoch eine längere Testfahrt voraus.

[22] Anzahl der Satelliten in Sichtweite.

[23] Formel zur Berechnung der Distanz ist beschrieben in Kapitel 3.3.5

[24] [S4] gibt hier 5 Meter mit Differential GPS und 15 Meter ohne DGPS an

Die Abweichungen in Messung zwei bis vier, acht und neun sind relativ hoch, da hier lange geradeaus Stücke, auf Autobahn und Bundesstraße, gefahren wurden. Der Karten-Abgleich ist in diesem Fall nicht so genau. Eine deutliche Verbesserung ist in den Messungen fünf bis acht zu erkennen. Hier bot die Fahrstrecke viele Knotenpunkte und Wendungen.

Abschließend soll noch verglichen werden, welcher Entfernung eine Abweichung von einer Sekunde, bzw. einer Minute bedeuten kann. Diese Werte sind nur für die jeweilige Lage gültig, da Längen- und Breitengrade, auf der Welt, unterschiedliche Abstände zu einander haben. An den Polen sind sie sehr nah zusammen, während sie am Äquator am weitesten auseinander liegen.

Abweichung	**Verändert in**	**Referenz Latitude/Longitude**	**Veränderung Latitude/Longitude**	**Delta in m**
1 Sek.	Latitude	50° 34‘ **5.0**“ N 8° 30‘ 20.0“ E	50° 34‘ **6.0**“ N 8° 30‘ 20.0“ E	30.92
1 Sek.	Longitude	50° 34‘ 5.0“ N 8° 30‘ **20.0**“ E	50° 34‘ 5.0“ N 8° 30‘ **21.0**“ E	19.64
1 Sek.	Latitude & Longitude	50° 34‘ **5.0**“ N 8° 30‘ **20.0**“ E	50° 34‘ **6.0**“ N 8° 30‘ **21.0**“ E	36.63
1 Min.	Latitude	50° **34**‘ 5.0“ N 8° 30‘ 20.0“ E	50° **35**‘ 5.0“ N 8° 30‘ 20.0“ E	1855.4
1 Min.	Longitude	50° 34‘ 5.0“ N 8° **30**‘ 20.0“ E	50° 34‘ 5.0“ N 8° **31**‘ 20.0“ E	1178.48
1 Min.	Latitude & Longitude	50° **34**‘ 5.0“ N 8° **30**‘ 20.0“ E	50° **35**‘ 5.0“ N 8° **31**‘ 20.0“ E	2197.91

Tabelle 2.2: Einfluß von Abweichung auf Entfernung

Abschließend bleibt festzustellen, dass die GPS Mouse und auch das Navigationssystem als GPS-Quellen geeignet erscheinen. Da mit dem Navigationssystem aber noch weitere Funktionen, wie zum Beispiel Zielführung, zur Verfügung stehen ist dessen Einsatz für diese Diplomarbeit von besonderem Interesse. Koordinaten können beispielsweise an das Navigationssystem gesendet werden, um in der Zielführung benutzt zu werden.

2.4 Text to Speech System

2.4.1 Das Festival Projekt

Für diese Arbeit wurde das Sprachsynthese System „Festival“, der University of Edinburgh[25], gewählt. Es handelt sich dabei um ein vielsprachiges Text to Speech System, das in C++ geschrieben wurde und einen Scheme[26] basierten Kommandointerpreter für allgemeine Kontrollen besitzt. Der Quellcode ist frei verfügbar und Möglichkeiten für Entwicklung und Forschung sind gegeben. Die Integration auf einem beliebigen Linux-System ist von einem geübten Benutzer leicht durchzuführen. Gute Unterstützung durch die Mailing-List „Festival-Talk“ und die generell kostenfreie Nutzung des TTS Systems rechtfertigen dessen Auswahl für diese Arbeit.

Unter den eingangs erwähnten Kriterien bezüglich der Auswahl eines Text to Speech Systems ist Festival für den Automotive Bereich allerdings nicht zu empfehlen. Der sehr hohe Memory Footprint von bis zu 40 MB, inklusive der Sprachen Deutsch und Englisch, ist eindeutig zu hoch und stößt damit bereits an die Grenzen der hier verwendeten Hardwareplattform. Auch die benötigte Rechenleistung von Festival liegt über den verfügbaren 47 MIPS, des verwendeten STPCs. Trotz diesem Mangel bei den technischen Eckdaten eignet sich Festival für die Darstellung der prinzipiellen Nutzung eines TTS Systems.

2.4.2 Schnittstellen zu Festival

Um auf die Sprachsynthese Funktionen von Festival zugreifen zu können gibt es eine Vielzahl von Schnittstellen. In der Festival Dokumentation wird hier von verschiedenen APIs[27] gesprochen. Da letztendlich alle APIs von einem Scheme Interpreter abhängig sind, wird dieser zuerst erwähnt. Er umfasst die komplette Programmiersprache Scheme und wird als die mächtigste API zu Festival bezeichnet.

Eine sehr einfach Methode, um Festival zu benutzen, ist die Shell API. Inhalte von Textfiles können direkt wiedergegeben werden. Als Beispiel nehmen wir an es gäbe das Textfile „**hallo.txt**“ mit dem Inhalt

Hallo Welt. Das ist ein Test von Festival.

Dann kann man Festival sehr einfach mit

unix$ festival –tts hello.txt

aufrufen und erhält eine entsprechende akustische Ausgabe.

Ausserdem kann Sable[28] markierter Text als Eingabeformat für Festival verwendet werden. Dies ist sehr ausführlich in Kapitel zehn der Festival Spezifikation [F1] beschrieben.

Um Festival in andere Programme einzubetten gibt es eine C/C++ und C only API. Java und JSAPI sind derzeit in der Entwicklung und nur als Initiale Versionen verfügbar. Gegenwärtig kommuniziert die JSAPI direkt mit dem Festival Server, anstatt als Teil des Java Prozesses.

Sehr ausführlich beschrieben ist die, in dieser Arbeit verwendete, Server/Client API. Hierbei handelt es sich um eine BSD[29] Socket basierte Schnittstelle. Somit ist es möglich, Festival als Server zu starten. Mehrere Clients können auf diesen Server zugreifen. Für jeden Client Vorgang wird eine neue Instanz des Festival Servers erzeugt. Es wird ausdrücklich darauf hingewiesen, dass diese Methode als unsicher gilt und möglicherweise beliebigen Benutzern Zugriff auf die ausführende Maschine gewährt. Dieses mögliche Problem wird bei dieser Arbeit nicht weiter betrachtet. Weitere Beschreibungen zur Server Zugangskontrolle, der Client Kontrolle und das Server/Client Protokoll sind in der Festival Dokumentation [F1] zu finden.

[25] Siehe [W14] für weitere Informationen

[26] Nichtprozedurale Programmiersprache, die zur Laufzeit übersetzt, also interpretiert, wird

[27] Application Programme Interface; Schnittstelle zu einem Programm

[28] Sable ist ein Satz von Schlüsseln und Symbolen die gesprochenen Text in TTS Systemen beschreiben. Es basiert auf der Extensible Markup Language (XML) und der Standard Generalized Markup Language (SGML).

[29] Berkeley Software Distribution bezieht sich auf die spezielle Version des UNIX Betriebssystems das von der University of California in Berkeley entwickelt und vertrieben wurde. Viele kommerzielle Implementierungen von UNIX Systemen basieren auf BSD.

2.5 Datenbank

Auf den Einsatz hoch entwickelter Datenbankprodukte wurde bereits im Vorfeld dieser Arbeit verzichtet. Trotzdem ist in Kapitel 3.2.6 ein Relationaler Datenbankentwurf angegeben, der Grundlage für die spätere Realisierung sein soll. Bedingt durch die Aufgabenstellung ist es notwendig, Informationen zu Sehenswürdigkeiten, gepaart mit Positionskoordinaten, in Textdateien abzulegen. Der Begriff Datenbank wird somit nur verwendet, um kenntlich zu machen, dass Informationen dauerhaft auf dem System gespeichert bleiben.

Dies wird in einfachen ASCII-Dateien realisiert, die mit Standardfunktionen (öffnen, lesen, schließen) bearbeitet werden können. Es werden verschiedene Dateiendungen eingesetzt, um Dateien in ihrer Bedeutung zu trennen. Inhalt der Dateien ist nach bestimmen Konventionen festgelegt worden und in Kapitel 3.3.7 beschrieben. Um den Zugriff auf Inhalte zu erleichtern, werden vorgefertigte Funktionen zum Parsen[30] von Textdateien verwendet.

---

[30] Aus dem Englischen übernommener Begriff der häufig anstelle von „Satzbau analysieren" verwendet wird.

3 Entwicklung des virtuellen Reiseführers

In den folgenden Kapiteln ist die tatsächliche Entwicklung des virtuellen Reiseführers beschrieben. Die Voruntersuchung des Begriffs „Reiseführer“ macht zunächst klar *was* implementiert werden soll. Es handelt sich also um eine einfache Betrachtung des verwendeten Begriffs, im Kontext der realen Welt. Ziel ist es einen existenten Reiseführer virtuell, mittels Software, nachzubilden.

Im darauffolgenden Kapitel „Softwareentwurf“ ist dargestellt *wie* eine solche Nachbildung realisiert werden könnte. Hierbei wird für die Softwareentwicklung eine objektorientierte Vorgehensweise, gegenüber anderen Methoden, vorgezogen. Verschiedene, aber nicht alle, Techniken der „Objekt-Orientierten Modellierung“[31] kommen zum Einsatz.
Nach dem Aufstellen möglicher Leistungsmerkmale, des virtuellen Reiseführers, folgt die Angabe sogenannter „Use Cases“. Die „Use Case Modellierung“ ist ein einfaches Hilfsmittel, um zu definieren was außerhalb des Computersystems existiert und welche Aufgaben durch das System erledigt werden sollen.
Um Use Cases besser zu verstehen werden, von der späteren Graphischen Benutzeroberfläche, sogenannte „Screen Prototypes“ [32] angefertigt. Zusätzlich werden noch Szenarien beschrieben, die nicht mit den Use Cases verwechselt werden sollen. Szenarien sind Instanzen eines Use Cases, ähnlich wie Objekte für Klassen.
Aus den vorhergehenden Bemühungen entsteht dann ein Klassenmodell und ein Datenbankentwurf, welche als Grundlage für die Implementierung dienen. Exakte Bestandteile der Implementierung sind im abschließenden Kapitel, zur Lösung der Aufgaben, angegeben.

3.1 Voruntersuchung des Begriffs Reiseführer

Bei dem Wort Reiseführer mögen dem Leser sofort zwei Dinge einfallen: Zum einen der Reiseführer in Form eines Buches und zum anderen der Reiseführer in Menschengestalt. Beide Vorstellungen sind absolut zulässig und Grundlage dieser Arbeit. Ein virtueller Reiseführer sollte in der Lage sein beide Formen auf einem Computersystem abzubilden. Die primären Merkmale beider Abbildungen werden kurz betrachtet und die virtuelle Synthese als Zielsetzung angegeben.

3.1.1 Mensch

Der Reiseführer, in Menschengestalt, steht im Allgemeinen nur während der Reise zur Verfügung und wird von Reisenden häufig erst vor Ort kennengelernt. Die Kommunikation zwischen Reisenden und Reiseführer erfolgt verbal und wird durch Gestik, Mimik oder technisches Gerät unterstützt. Ein Reisender erwartet, dass er zu den schönsten und interessantesten Plätzen hingeführt wird. Dort werden Informationen durch den Reiseführer punktgenau und individuell vermittelt.

3.1.2 Buch

In Buchform bietet sich ein Reiseführer zur Vorbereitung, während der Reise und auch nach der Reise an[33]. Um Informationen aus einem Buch zu erlangen, muß es allerdings gelesen, oder zumindest angeschaut werden. Reiserouten und das Besichtigen von Sehenswürdigkeiten kann individuell gestaltet werden. Oftmals werden Tourvorschläge gemacht, aber von einer Führung im eigentlichen Sinne kann nicht gesprochen werden. Ausserdem bietet ein solcher Reiseführer einen enormen Hintergrund an Daten, die vom Leser abgerufen werden können. Der Ort an dem sich der Reisende während dessen befindet spielt eine untergeordnete Rolle.

---

[31] Etliche Methoden der Objekt-Orientierten Modellierung haben sich im Laufe der Jahre entwickelt. Dabei haben Autoren immer wieder auf Ideen und Methoden anderer aufgebaut. Für diese Arbeit wird keine spezielle Ausprägung der Objekt-Orientierten Entwicklung vorgeschrieben. Als unterstützende Literatur seien lediglich [B1] und [B2] angegeben.

[32] Aus dem Englischen übernommener Begriff, der mit „Bildschirm Muster“ übersetzt werden kann.

[33] Zum bewusst machen des Begriffs wurden [B3] und [B4] verwendet.

3.1.3 Virtuelle Synthese – Zielsetzung

Wie eingangs erwähnt sollte ein virtueller Reiseführer in der Lage sein, beide Abbildungen auf einmal zu realisieren. Man kann ihn derart gestalten, dass er zur Reisevorbereitung, während der Reise und nach der Reise präsent ist. Mittels Sprachsynthese und Spracherkennung[34] kann eine verbale Kommunikation mit dem Benutzer stattfinden, wobei Gestik und Mimik durch ein Display simuliert werden könnten. Informationen zu Sehenswürdigkeiten können ebenfalls auf einem Display dargestellt und vom Benutzer gelesen oder einfach angeschaut werden. Sicherlich kann ein Reisender mit seinem Automobil die Fahrtroute selber bestimmen oder aber durch ein angeschlossenes Navigationssystem geführt werden. Durch exakte Positionsbestimmung ist es auch möglich, Informationen punktgenau und individuell wiederzugeben. Der Hintergrund an Daten ist abhängig vom Inhalt, der auf dem System platzierten, Datenbank.

3.2 Softwareentwurf

In den folgenden Kapiteln sind Ergebnisse des erstellten Softwareentwurfs dargestellt. Hierbei wurden, wie oben erwähnt, verschiedene Techniken der Objekt-Orientierten Modellierung verwendet. Zu jeder verwendeten Technik gibt es ein eigenes Unterkapitel und eine einführende Beschreibung der Methode.

3.2.1 Mögliche Leistungsmerkmale

Aus der Voruntersuchung des Begriffs Reiseführer wurden Leistungsmerkmale abgeleitet, die tatsächlich implementiert werden können. Diese sind nachfolgend aufgelistet und als Möglichkeit zu verstehen. Nicht alle Merkmale können im Rahmen dieser Arbeit implementiert werden.

- Browser-basierter Reiseführer
  - Abbildung des Reiseführers in Buchform
  - Passagiere im Fond, Beifahrer oder Fahrer (im Stand) können HTML-Dateien browsen
  - Von Bedeutung sind Übersichtlichkeit, Hintergrund und Service
- Verbale Reiseinfo
  - Informationen werden abhängig von der aktuellen Fahrzeugposition ausgegeben
  - Sprachsynthese kommt zum Einsatz
  - Benutzung des Systems über Fernbedienung
- Tourenführung
  - Ziel- und Wegpunkte stammen vom Reiseführer
  - Zielführung durch angeschlossenes Navigationssystem
- Aktualisieren der Datenbank
  - Die Datenbank kann auf Dateiebene bearbeitet (Hinzufügen, Anzeigen, Entfernen) werden
  - Datenbankprodukte sind in verschiedene Kategorien, wie z.B. Land, Region, Stadt und Tour eingeteilt

---

[34] Spracherkennung ist nicht Gegenstand dieser Arbeit und zum Zeitpunkt nicht auf der Hardwareplattform verfügbar.

3.2.2 Use Cases

Die Modellierungstechnik der Use Cases ist Bestandteil der meisten Objekt-Orientierten Methoden. Das „Use Case Modell“ kennt **Akteure** und **Use Cases**.

Ein **Akteur** ist alles außerhalb des Systems, das mit diesem in Kontakt tritt. Es kann sich hierbei um eine Person, eine Gruppe von Personen, eine Organisation, ein weiteres System oder ein Gerät handeln. Da Akteure außerhalb des Systems sind hilft deren Identifikation beim finden der Systemgrenzen.

Der **Use Case** ist eine Abfolge von verhaltensgleichen Interaktionen die durch einen Akteur, im Dialog mit dem System, durchgeführt werden. Dabei kommt dem Akteur ein meßbarer Nutzen zu gute.

Das vorherige Bestimmen möglicher Leistungsmerkmale ist hilfreich beim Aufstellen der Use Cases[35], die für diese Arbeit gültig sein sollen. In der folgenden Abbildung sind Use Cases und entsprechende Akteure in einem Diagramm dargestellt und nachfolgend beschrieben.

Beim Softwareentwurf mittels Use Cases tauchen erstmals die Begriffe „Gesamtsystem“ und „Backbone“ auf. Sie werden ausschließlich in diesem Kapitel verwendet um die mögliche Einbindung des virtuellen Reiseführers in ein kommerzielles Gesamtkonzept zu verdeutlichen. Über einen sogenannten Backbone wäre es Benutzern möglich, Datenbankinhalte auf das MC5400 nachzuladen. Ein entsprechendes Preismodell könnte finanziell interessant sein, ist aber nicht Gegenstand dieser Arbeit. Die Use Cases und Akteure des Teilbereichs „Backbone“ werden zwar beschrieben, aber im weiteren Verlauf nicht konkretisiert.

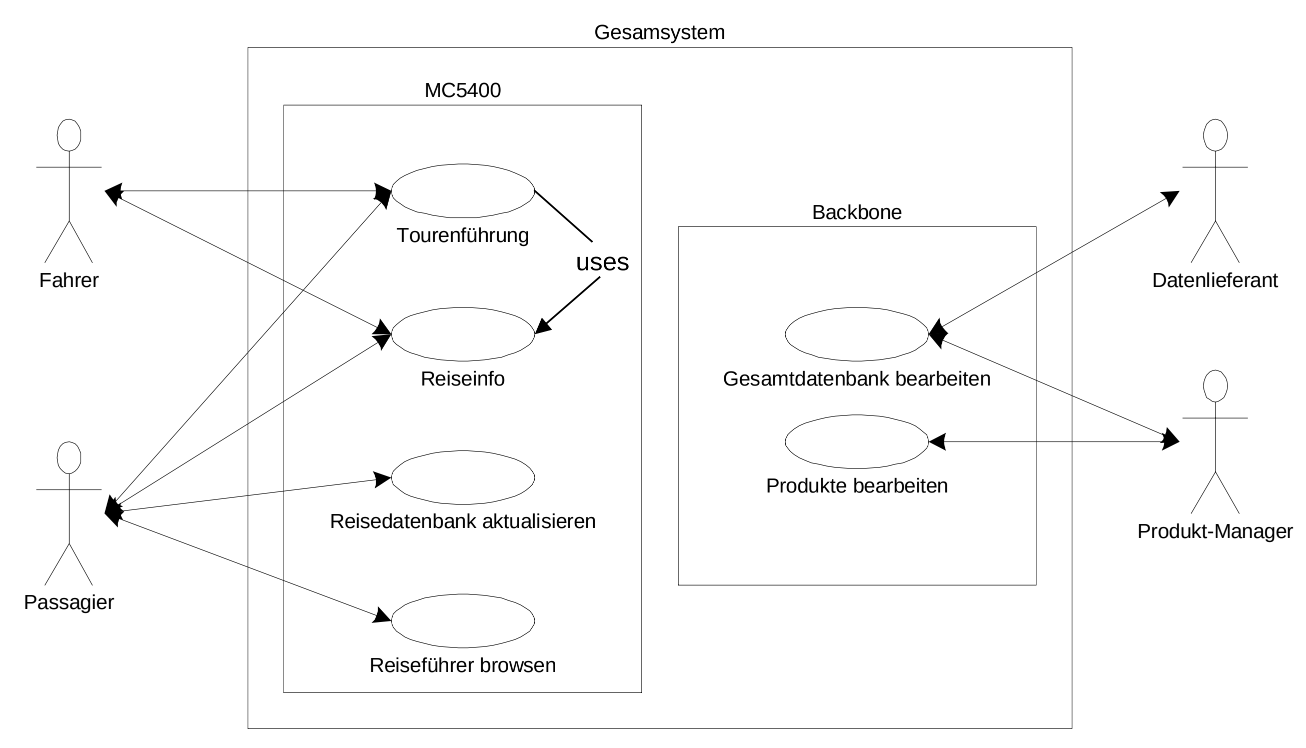


Abbildung 3.1: Use Cases und Akteure des virtuellen Reiseführers

Um die Use Cases untereinander in ihrer Schwierigkeit bewerten zu können wird ein subjektives Bewertungssystem eingesetzt. Fünf Schwierigkeitsgrade seien wie folgt beschrieben:

1) **Leicht**, Standardaufgabe die als Voraussetzbar angesehen werden kann.
2) **Gehoben**, verlangt Systemkenntnisse und geht über die Standardaufgabe hinaus.
3) **Aufwendig**, ist eine Aufgabe des Grades zwei, die zusätzlich zeitintensiv ist.
4) **Komplex**, weitere Zerlegung in Teilaufgaben notwendig, dennoch in Einzelarbeit zu leisten.
5) **Sehr komplex**, ist eine Aufgabe des Grades vier, zeitintensiv und sollte im Team gelöst werden.

---

[35] Eine Technik der Objekt Orientierten Modellierung

3.2.2.1 Use Cases im Bereich MC5400

Name:	Tourenführung
Bedingung:	Verbindung zu einem Navigationssystem via CCI.
Benutzt von:	Fahrer, Passagier
Beschreibung:	Durch den Fahrer, oder einen der Passagiere kann die Tourenführung bedient werden. Der Benutzer wählt eine Tour aus den kategorisierten Tourvorschlägen aus. Entsprechende Daten werden aus der Reise-Datenbank geladen. Auf einer Tour befinden sich typischerweise mehrere Sehenswürdigkeiten, die als Wegpunkte in der Reise-Datenbank verfügbar sind. Die Zielkoordinaten eines jeden Wegpunktes werden dem Navigationssystem übermittelt und angefahren. Erst nach erreichen, oder löschen des aktuellen Wegpunktes wird der nächste übermittelt und angefahren. Reiseinformationen werden eigenständig während der Fahrt, oder bei erreichen eines Wegpunktes, durch den Reiseführer ausgegeben.
Zweck:	Den Reisenden auf einer Tour führen und während dieser Reiseinformationen auszugeben.
Hinweis:	Auf dem Monitor des Fahrers wird die MMI des Navigationssystems dargestellt. Für die Passagiere im Fond kann eine anderweitig genutzte Grafikausgabe erfolgen. Zum Beispiel das Tourguide browsen, oder aber grafische Hinweise zur Tourenführung (Pfeile, gezeichnete Figur, etc.).
Bewertung:	**4** Die Tourenführung wird als komplexer Anwendungsfall betrachtet. Die richtige Kommunikation mit dem Navigationssystem, in Abhängigkeit zur Reise-Datenbank, ist eine Herausforderung. Die Synchronisation der Geokoordinaten ist ein eigenes Teilproblem.

Tabelle 3.1: Use Case „Tourenführung“

Name:	Reiseinfo
Bedingung:	Mindestens eine angeschlossene GPS-Quelle
Benutzt von:	Fahrer, Passagier
Beschreibung:	Die Reiseinfo kann durch den Fahrer, oder einen der Passagiere bedient werden. Bei der Reiseinfo handelt es sich um eine Grundfunktion der Tourenführung. Allerdings wird der Benutzer hier nicht durch ein Navigationssystem geführt. Die Route kann individuell gewählt werden. Orte an denen die Ausgabe einer Reiseinformation erfolgt werden dementsprechend nur zufällig erreicht. Trotzdem gibt das System diese eigenständig während der Fahrt aus. Es wäre allerdings möglich dem Benutzer, Browserbasiert, Touren vorzuschlagen, die er dann mittels Karte, oder Ortskenntnis selber anfahren muß. Das erhöht zumindest die Wahrscheinlichkeit der Ausgabe von Reiseinformationen.
Zweck:	Dem Reisenden während der individuellen Fahrt Informationen auszugeben.
Bewertung:	**3** Bei der Reiseinfo sind ähnliche Probleme zu lösen, wie bei der Tourenführung, dennoch nicht so anspruchsvoll.

Tabelle 3.2: Use Case „Reiseinfo“

Name:	Reisedatenbank aktualisieren
Bedingung:	Internet-Konnektivität und ausreichend freier Flash-Speicher
Benutzt von:	Passagier
Beschreibung:	Ein Passagier kann per Download die Reisedatenbank aktualisieren. Hierfür sind zwei Methoden vorgesehen: 1.) Browser basiert, mit Zugriffsbeschränkung auf bestimmte Dateitypen 2.) Verwendung eines dedizierten Download-Clients Möglich soll es sein Datenbündel entsprechend einer der Kategorien Land, Region, Stadt oder Tourenvorschlag herunter zu laden.
Zweck:	Den Reisenden mit neuen Produkten zu versorgen.
Hinweis:	Realisierung ist stark abhängig vom Projektfortschritt des Zielsystems
Bewertung:	**2** Das Aktualisieren einer Datenbank via Internet kann grundsätzlich als Standardaufgabe betrachtet werden (also mit **1** zu werten). Systemaspekte sind aber derzeit noch unklar, deswegen die höhere Bewertung.

Tabelle 3.3: Use Case „Reisedatenbank aktualisieren“

Name:	Reiseführer browsen
Bedingung:	Optionale Verbindung zu einem Navigationssystem
Benutzt von:	Passagier
Beschreibung:	Sämtliche Daten die in der Reisedatenbank vorhanden sind werden Browsergerecht aufbereitet. Ein Passagier kann über „Links“ Reiseinformationen abrufen. Außerdem ist eine Suchfunktion einzurichten, die unter Eingabe von Schlagworten die entsprechenden Sehenswürdigkeiten auflistet. Nähere Informationen zu den Sehenswürdigkeiten können angezeigt werden. Besteht eine Verbindung zu einem Navigationssystem so kann die Zielführung für das entsprechende Objekt aktiviert werden.
Zweck:	Betrachten von Reiseinformationen, unabhängig von der Fahrzeugposition.
Bewertung:	**4** Das Browsergerechte aufbereiten der Reisedatenbank wird in erster Linie aufwendig sein. Die Schwierigkeit besteht im festlegen richtiger Strukturen und dem Schaffen einer anspruchsvollen Benutzerschnittstelle. Da aber noch weitere Features, wie z.B. „Suchfunktion“ beinhaltet sein können, gestaltet sich dieser Anwendungsfall auch eher komplex.

Tabelle 3.4: Use Case „Reiseführer browsen“

3.2.2.2 Akteure im Bereich MC5400

Name:	Fahrer
Bedingung:	Das Fahrzeug bewegt sich. Im Stand ist der Fahrer in der Rolle eines Passagiers.
Use Case:	Tourenführung, Reiseinfo
Beschreibung:	Der Fahrer ist im Allgemeinen die Person die das Fahrzeug steuert. Alle Aktionen die der Fahrer ausführt müssen während dem Betreiben eines Fahrzeuges möglich sein. Beispielsweise ist dem Fahrer das Fernsehen nicht erlaubt. Somit ergibt sich auch selbstverständlich das Reiseführer browsen oder aktualisieren der Reisedatenbank keine Funktionen sind, die der Fahrer ausführen sollte. Es sei ihm aber gestattet die Tourenführung bzw. Reiseinfo während der Fahrt zu bedienen. Dialogbasiert können verschiedene Funktion, wie zum Beispiel aktivieren/beenden der Führung oder wiederholen einer Ansage, ausgeführt werden.

Tabelle 3.5: Akteur „Fahrer“

Name:	Passagier
Bedingung:	Diese Person befindet sich nicht auf dem Fahrersitz.
Use Case:	Tourenführung, Reiseinfo, Reiseführer browsen, Reisedatenbank aktualisieren
Beschreibung:	In der Rolle des Passagiers befinden sich die Personen im Fahrzeug, die es nicht führen. Im wesentlichen ist das der Beifahrer und die Personen im Fond. Einem Passagier stehen sämtliche Funktionen des Systems, auch während der Fahrt, zu Verfügung.
Hinweis:	Hierbei ist eine Einschränkung bezüglich der Grafikausgabe vorzunehmen. Die Darstellung von Geräten mit eigener Grafikausgabe, wie z. B. Navigation oder TV, kann beliebig auf die Monitore geschaltet werden. Darstellungen des MC5400 können aber immer nur einheitlich sein, da nur eine Grafikkarte zur Verfügung steht. Es ist also nicht möglich dem Fahrer, auf seinem Monitor, Darstellungen zur Tourenführung/Reiseinfo zu bieten und gleichzeitiges Benutzen der Funktion „Reiseführer browsen“ durch einen Passagier im Fond.

Tabelle 3.6: Akteur „Passagier“

3.2.2.3 Use Cases im Bereich Backbone

Name:	Gesamtdatenbank bearbeiten
Bedingung:	Definierte Schnittstellen
Benutzt von:	Datenlieferant, Produkt-Manager
Beschreibung:	Inhalte der Gesamtdatenbank können hier bearbeitet werden. Es ist möglich neue Inhalte einzustellen, bestehende Daten zu aktualisieren, oder aber auch zu löschen. Das verändern der allgemeinen Datenstruktur ist nicht vorgesehen. Die beiden zugewiesenen Rollen, Datenlieferant und Produkt-Manager, sollten Änderungen an der Gesamtdatenbank nur in Abstimmung erledigen. Der Produkt-Manager stellt hierbei eine kontrollierende Instanz dar.
Zweck:	Haushalten der gesamten Datenbestände.
Bewertung:	**5**
	Ist eine sogenannte „Backbone-Anwendung“. Know-how und Ressourcen spielen bei realer Implementierung eine besonders große Rolle. Um die Kleinstsimulierung eines solchen Systems zu erstellen sollte jedoch die Bewertung **3** ausreichen.

Tabelle 3.7: Use Case „Gesamtdatenbank bearbeiten“

Name:	Produkte bearbeiten
Bedingung:	Definierte Schnittstellen
Benutzt von:	Produkt-Manager
Beschreibung:	Aus den Inhalten der Gesamtdatenbank werden kommerzielle Produkte, also Untermengen des Gesamtinhaltes, geschaffen. Es kann festgelegt werden über welches Medium ein Produkt verfügbar gemacht wird und zu welchem Preis. Bei den Medien gäbe es zunächst die Unterscheidung zwischen Internet und einem physikalischen Datenträger.
Zweck:	Einen Teil der Datenbank auswählen und daraus ein Produkt schaffen.
Bewertung:	**3**
	Kann wiederum als Teillösung des Anwendungsfalls „Gesamtdatenbank bearbeiten" angesehen werden.

Tabelle 3.8: Use Case „Produkte bearbeiten"

3.2.2.4 Akteure im Bereich Backbone

Name:	Datenlieferant
Bedingung:	Definierte Schnittstellen
Use Case:	Gesamtdatenbank bearbeiten
Beschreibung:	Der Datenlieferant beschäftigt sich mit dem Sammeln, Katalogisieren und Auswählen geeigneter Daten, für die Anwendung des Reiseführers. Eine entsprechende Geo-Kodierung von Reisezielen muß bereits durch den Datenlieferant gewährleistet sein. Als Datenlieferant können auch interne Abteilungen von Siemens VDO auftreten. Es werden aber typischerweise externe Partner sein, die sich vornehmlich mit dieser Thematik beschäftigen.
Hinweis:	Die Gesamtdatenbank ist die Obermenge bezüglich der Reisedatenbank und enthält sämtliche verfügbaren Informationen. Eine Reisedatenbank, auf dem MC5400, enthält immer nur einen individuellen Ausschnitt der Gesamtdatenbank.

Tabelle 3.9: Akteur „Datenlieferant"

Name:	Produkt-Manager
Bedingung:	Definierte Schnittstellen
Use Case:	Gesamtdatenbank bearbeiten, Produkte bearbeiten
Beschreibung:	Der Produkt-Manager ist Mitarbeiter von Siemens VDO und zumeist in einer am Markt orientierten Abteilung tätig. In Abstimmung mit dem Datenlieferant sorgt er für das korrekte Einpflegen neuer Datenbestände in die Gesamtdatenbank. Aus der ihm verfügbaren Gesamtdatenbank kann er Produkte definieren. Diese Produkte können auf der Granularität Land, Region und Stadt oder Tourenvorschlag basieren. Ein Benutzer von des Reiseführers kann, möglicherweise gegen Entgelt, seine Reisedatenbank durch das herunterladen von Produkten aktualisieren.
Hinweis:	Beachte Unterschied zwischen Reisedatenbank und Gesamtdatenbank.

Tabelle 3.10: Akteur „Produkt-Manager"

3.2.3 Screen Prototypes

Vor der realen Implementierung eines Computerprogramms hat sich die Arbeit mit Screen Prototypes als nützlich erwiesen. Sie helfen Entwickler und Anwender sich auf ein einheitliches Interface und den möglichen Programmablauf zu verständigen. Solche Prototypen und können ohne jegliche Programmierkenntnis erstellt werden. Auch für diese Arbeit wurden die folgenden Screen Prototypes erstellt.

Das Hauptmenü des MC5400 beinhaltet eine Reihe von Icons die den Zugang zu verschiedenen Applikationen ermöglichen. Für den virtuellen Reiseführer wird ein neues Icon „GoTo" eingefügt, über das ein entsprechendes Untermenü erreicht werden kann.

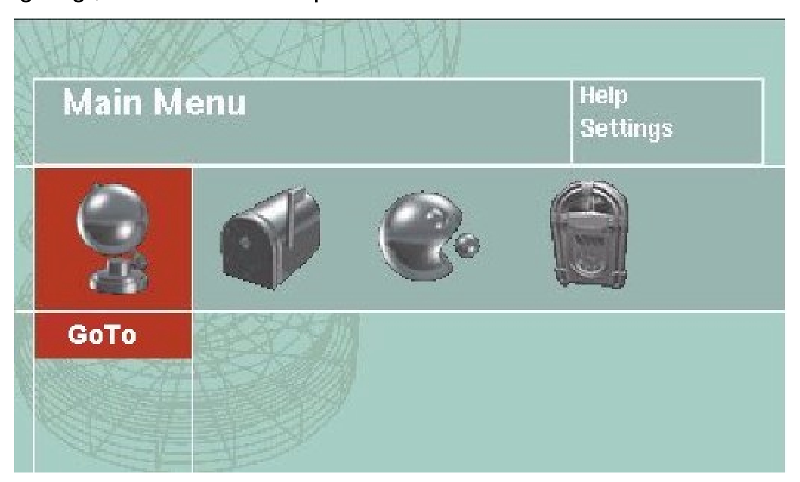


Abbildung 3.2: Screen Prototype „Main Menu"

Im ersten Untermenü des virtuellen Reiseführers *GoTo* finden sich fünf Einträge die aus den Use Cases wieder erkannt werden können. Auf der linken Seite sind die Menüpunkte aufgelistet. Rechts davon befindet sich ein eventueller Status bezüglich des Menüpunktes.

Nach dem Aufstarten befindet sich die Reiseinfo generell im Modus „inaktiv". Die Tourenführung ist beendet und zeigt „keine Tour gewählt". Zu den Einträgen „Browsen", „Datenbank" und „Über *GoTo*" gibt es keine Status Informationen.



Abbildung 3.3: Screen Prototype „Virtueller Reiseführer *GoTo*"

Mittels Fernbedienung kann die Reiseinfo aktiviert werden. Steht keine Sprachausgabe und keine geeignete GPS-Quelle zur Verfügung wird die Reiseinfo auf „inaktiv“ zurückgesetzt. Ist die Reiseinfo aktiv, so wird das durch den entsprechenden Status angezeigt. Akustische Ausgaben zu Sehenswürdigkeiten erfolgen spontan, wenn eine Sehenswürdigkeit im Umkreis festgestellt wurde.



Abbildung 3.4: Screen Prototype „Reiseinfo bedienen“

Die Tourenführung bietet Möglichkeiten eine Tour aus- bzw. abzuwählen. Wurde eine Tour ausgewählt kann diese gestartet, oder gestoppt werden. Eine gestartete Tour veranlasst die Zielführung zu deren Sehenswürdigkeiten, mittels Navigationssystem. Wird eine Tour nur ausgewählt und nicht gestartet, so befindet sie sich im Status „gestoppt“.



Abbildung 3.5: Screen Prototype „Tourenführung bedienen“

Eine Tour wird über ein entsprechendes Untermenü ausgewählt.



Abbildung 3.6: Screen Prototype „Tour auswählen“

Durch aktivieren des Menüpunktes „Browsen“ wird ein HTML-Browser geöffnet. Dieser ermöglicht die Darstellung von Produkten, die in der Reisedatenbank des Systems enthalten sind. Dieser Zweig wird hier jedoch nicht weiter verfolgt.
In den Statusanzeigen ist zu sehen, dass die Reiseinfo aktiv ist und die „Hessen-Tour“ für die Tourenführung ausgewählt wurde. Im Moment ist diese allerdings gestoppt.



Abbildung 3.7: Screen Prototype „Browsen“

Drei verschiedene Operationen sind auf die Inhalte der Datenbank anwendbar. Mittels „Hinzufügen“ kann eine Datenbankdatei auf das System nachgeladen werden. Entfernen löscht eine auszuwählende Datenbankdatei. Über das Feld „Anzeigen“ wird ein weiteres Menü geöffnet, in dem eine Datenbankdatei zur Anzeige ausgewählt werden kann. Zur Anzeige kommt ein HTML-File bezüglich der ausgewählten Datenbankdatei, welches eine Beschreibung der Datei enthält.



Abbildung 3.8: Screen Prototype „Datenbank bedienen“

Die Auswahlliste für das Anzeigen einer Datenbankdatei enthält alle in der Datenbank bekannten Produkte. Vergleiche hierzu Abbildung 3.6, wo lediglich die als Tour gekennzeichneten Datenbankdateien aufgeführt sind.



Abbildung 3.9: Screen Prototype „Datenbankdatei auswählen“

Unter den Menüpunkt „Über *GoTo*“ findet der Benutzer eine kurze Information über den virtuellen Reiseführer. Ausführliche Bedienungshinweise werden einheitlich in der Hilfefunktion des MC5400 abrufbar sein.

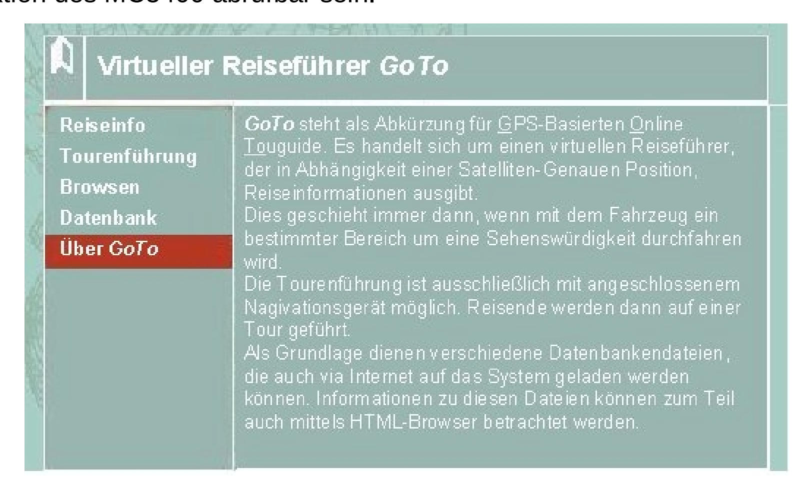


Abbildung 3.10: Screen Prototype „Über *GoTo*“

3.2.4 Szenarien

Um ein noch besseres Gefühl für die spätere Verhaltensweise des Reiseführers zu bekommen werden verschiedene Szenarien beschrieben. Die im folgenden beschriebene Szenarien stellen einen Bezug zu den Use Cases her. Sind allerdings nur als Beispiele zu betrachten.

Name:	Autobahn
Use Case:	Reiseinfo
Bedingung:	Reiseinfo aktiv
Beschreibung:	Bei einer Fahrt auf der Autobahn ist nicht anzunehmen, dass Reisende in unmittelbarer Nähe einer Sehenswürdigkeit vorbei fahren. In diesem Fall wird der Reiseführer, beim Passieren einer größeren Stadt, eine dort befindliche Sehenswürdigkeit ansagen. Ähnlich wie touristische Hinweisschilder auf der Autobahn. Dem Reisenden wird, unter Nennung der Adresse, vorgeschlagen in die Stadt abzufahren und die Sehenswürdigkeit zu besichtigen. Bei angeschlossenem Navigationssystem kann eine Zielführung angeboten werden. Wird das Ziel erreicht, möglicherweise Interaktion mit dem Benutzer nötig, können weitere Ansagen erfolgen.
Hinweis:	Basiert auf einem speziellen Produkt „Autobahn“. Geokoordinaten des Autobahnteilstücks und der Sehenswürdigkeit müßten hinterlegt sein. Das herunterladen einer detaillierten Datenbank zur entsprechenden Stadt sollte vom System motiviert werden.

Tabelle 3.11: Szenario „Autobahn“

Name:	Allgemeine Reiseinformationen
Use Case:	Reiseinfo
Bedingung:	Reiseinfo aktiv
Beschreibung:	Der Benutzer hat die Funktion Reiseinfo aktiv geschaltet. Aus Sicht des Benutzers wird der Reiseführer spontan Reiseinformationen ausgeben. Dies geschieht immer dann wenn ein Bereich von Geokoordinaten durchfahren wird, für den es einen Datenbankeintrag gibt. Auf dem Bildschirm erscheint eine Ansagefigur und der entsprechende Text wird per Text to Speech vorgelesen.

Tabelle 3.12: Szenario „Allgemeine Reiseinformationen“

Name:	Allgemeine Tourenführung
Use Case:	Tourenführung, Reiseinfo
Bedingung:	Tourenführung gestartet, Reiseinfo aktiv
Beschreibung:	Der Reisende wählt aus einer Liste von Touren eine bestimmte aus. Nach der Auswahl startet er die Tourenführung. Die Geokoordinaten des ersten Ziels, auf der Tour, werden an das Navigationssystem übertragen und die Zielführung wird gestartet. Auf dem Weg zum Ziel kann es zu spontanen Ausgaben durch die Reiseinfo kommen. So wird auch die Ansage zur Sehenswürdigkeit am Zielpunkt, ganz automatisch, erfolgen. Die Steuerung der Tourenführung fragt jedoch beim Benutzer nach ob weitere Informationen, falls vorhanden, zum Ziel vorgelesen werde sollen. Oder ob die nächste Sehenswürdigkeit, auf der Tour, als Ziel gesetzt werden soll. Der Reisende entscheidet sich dafür, die nächste Sehenswürdigkeit zu überspringen und die Tour mit dem darauf folgenden Ziel fort zu setzen.

Tabelle 3.13: Szenario „Allgemeine Tourenführung“

3.2.5 Klassenmodell

Da für die Implementierung dieser Arbeit die Objektorientierte Sprache C++[36] gewählt wurde, macht es Sinn zunächst ein Klassenmodell anzugeben. In der folgenden Abbildung sind die zu implementierenden Klassen dargestellt, jede Klasse wird zusätzlich noch in den folgenden Kapiteln beschrieben. Attribute und Methoden wurden aus Gründen der Übersichtlichkeit in der Darstellung weggelassen, sind dafür aber im Anhang A aufgeführt.

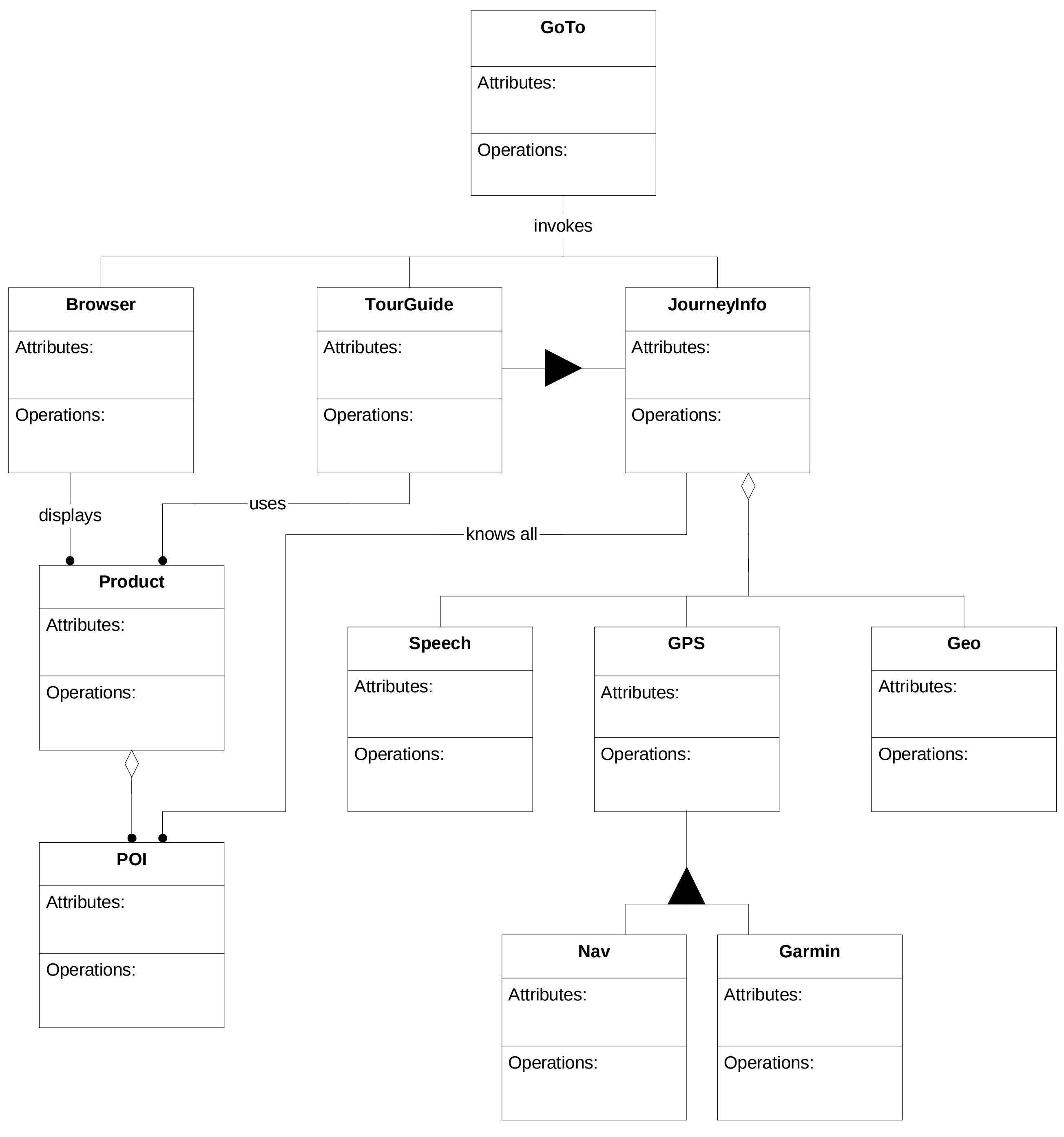


Abbildung 3.11: Klassenmodell des Reiseführers

36 Als Programmierreferenz wurde [B5] verwendet.

3.2.5.1 Klasse GoTo

Die Klasse „GoTo“ übernimmt hauptsächlich Verwaltungsaufgaben und kann als Interface Klasse zur MMI des MC5400 angesehen werden. Im Bezug auf den Reiseführer übernimmt sie Funktionalitäten im Bereich der MMI und der Initialisierung weiterer Komponenten. Leistungsmerkmale wie die „Reiseinfo“, „Reiseführer Browsen“, oder die „Reiseführung“ werden über die Klasse GoTo aufgerufen.

3.2.5.2 Klasse Product

Durch die Klasse „Product“ werden Objekte der realen Welt, von Reiseführern in Buch Form, repräsentiert. Aufgrund der Zusammensetzung aus sogenannten POI’s (Points Of Interest) kann im Prinzip jeder beliebige Reiseführer realisiert werden. Hier ist es allerdings nur möglich mit Instanzen eines vorher bestimmten Produkts zu arbeiten. Neue Produkte können im System nicht erstellt werden, sie werden aus der Datenbank gelesen.

3.2.5.3 Klasse POI

Jeder Punkt der als Sehenswürdigkeit, oder anderweitig, für den Reiseführer von Interesse ist wird als POI abgelegt. Ein POI wird nicht durch den Reiseführer erzeugt, sondern von ihm genutzt. Er wird bei Bedarf aus der Datenbank gelesen. Es ist notwendig jeden POI einer Instanz der Klasse Product zuzuordnen. So kann ein POI im System eindeutig referenziert werden.

3.2.5.4 Klasse Browser

Mit der Klasse „Browser“ wird die Verbindung zu einem HTML-Browser hergestellt. In der Regel wird eine anzuzeigende Datei übergeben. Über die Produkte gelangt ein Objekt der Klasse Browser an enthaltene POI’s. Soll ein POI zur Anzeige gebracht werden, so werden seine Attribute in einer HTML-Datei aufbereitet. Diese wird dann durch den HTML-Browser, auf dem Display dargestellt.

Dafür ist kein Wissen über die aktuelle Position des Fahrzeugs notwendig. Für den Fall, dass ein Navigationssystem angeschlossen ist, kann jedoch die Zielführung zu einem ausgewählten POI aktiviert werden.

3.2.5.5 Klasse JourneyInfo

Durch die „JourneyInfo“ wird das Leistungsmerkmal „Reiseinfo“ realisiert. Der Weg des Reisenden wird kontinuierlich verfolgt. Die aktuelle Position des Fahrzeugs wird mit allen Positionen von POI’s verglichen, die sich auf dem System befinden. Für den Positionsvergleich können verschiedene Methoden eingesetzt werden. Wird ein Vergleich als erfolgreich bewertet, so gibt die Sprachausgabe den beschreibenden Text des POIs wieder. Weitere Aktionen sind von der Art des POIs und Handlungen des Benutzers abhängig.

3.2.5.6 Klasse TourGuide

Der „TourGuide“ ist eine Spezialisierung von JourneyInfo. Der Weg des Reisenden wird als Tour vorgegeben. Falls sich spezielle Datenbankdateien des Typs „Tour“ auf dem System befinden, wählt der Benutzer zunächst eine dieser Touren aus. Bei angeschlossenem Navigationssystem werden Zielkoordinaten vom System an die Navigation übertragen. Auf der Tour befindliche POI’s werden nacheinander abgefahren. Ausgabe und Behandlung der POI’s erfolgt nach den Methoden von JourneyInfo.

3.2.5.7 Klasse Speech

„Speech“ ist die Klasse, die eine Schnittstelle zum eingesetzten Text to Speech System bildet. Allgemeine Initialisierungen der Sprachausgabe und Einstellen der verwendeten Sprache können hier gemacht werden. Ausserdem stehen Funktionen zur Verfügung, um während der Laufzeit Veränderungen am Text to Speech System vorzunehmen. Die Funktion der tatsächlichen Sprachausgabe wird hier gekapselt und auszugebender Text wird an das Text to Speech System weitergegeben.

3.2.5.8 Klasse Geo

Die Klasse „Geo“ liefert geometrische Funktionen für den virtuellen Reiseführer. Datentypen bezüglich Geokoordinaten, für die Positionsbestimmung, sind hier einheitlich definiert. Funktionen für die Umrechnung von Koordinaten, Abstandsberechnung von Positionen und Positionsvergleiche sind in dieser Klasse zusammengefasst.

3.2.5.9 Klasse GPS

„GPS“ gruppiert die verschiedenen Alternativen von GPS-Quellen zusammen. Es ist möglich den Reiseführer entweder mit einem angeschlossenen Navigationssystem (über CCI) oder einem handelsüblichen GPS Empfänger, wie die GPS Mouse von Garmin, zu betreiben.

Das Auffinden einer geeigneten GPS-Quelle ist Hauptaufgabe dieser Klasse. Zu diesem Zweck sind Grundfunktionalitäten (z.B. Initialisierung und Verbindungsaufbau) jeder möglichen GPS-Quelle hier realisiert. Individuelle Implementierungen bezüglich einer GPS-Quelle wird dann in einer entsprechenden Klasse gemacht. Derzeit gibt es die Klassen „Nav“, für das Navigationssystem und „Garmin“, für die GPS Mouse von Garmin.

3.2.5.10 Klasse Nav

Die Funktionen der Klasse „Nav“ bilden das Interface zum Navigationssystem. Sie basieren auf den Vorgaben des CCI Kommunikationsprotokolls. Es können Konfigurationen an der Art der Datenübertragung vorgenommen werden. Telegramme für die Positionsbestimmung und das setzen neuer Zielkoordinaten werden mit dem Navigationssystem ausgetauscht.

3.2.5.11 Klasse Garmin

Stellt das Interface zu Garmin’s GPS Mouse her, ähnlich wie Nav zum Navigationssystem. Ein Objekt von Garmin liest kontinuierlich die von der GPS Mouse gesendeten Daten. Diese werden analysiert und aufbereitet. In der Regel werden alle Sätze die eine Positionsbestimmung beinhalten berücksichtigt.

3.2.6 Datenbankentwurf

Als Grundlage für den Reiseführer werden POI’s, gruppiert in Produkten, benötigt. Im wesentlichen handelt es sich dabei um permanente Instanzen der entsprechenden Klassen Product und POI. Auch wenn für diese Arbeit der Einsatz eines Relationalen Datenbanksystems ausgeschlossen wurde, so soll zumindest ein Entwurf angegeben werden. Dieser hilft beim Verstehen der später tatsächlich realisierten Datenbankdateien.

In Abbildung 3.12 sind drei Tabellen dargestellt die in ihrer Relation den Datenbankentwurf wieder geben. Die Tabellen Product und POI sind konkrete Abbilder der entsprechenden Klassen. Quick Reference wird nur für Positionsvergleiche verwendet und ist später innerhalb der Klasse „JourneyInfo“ realisiert.

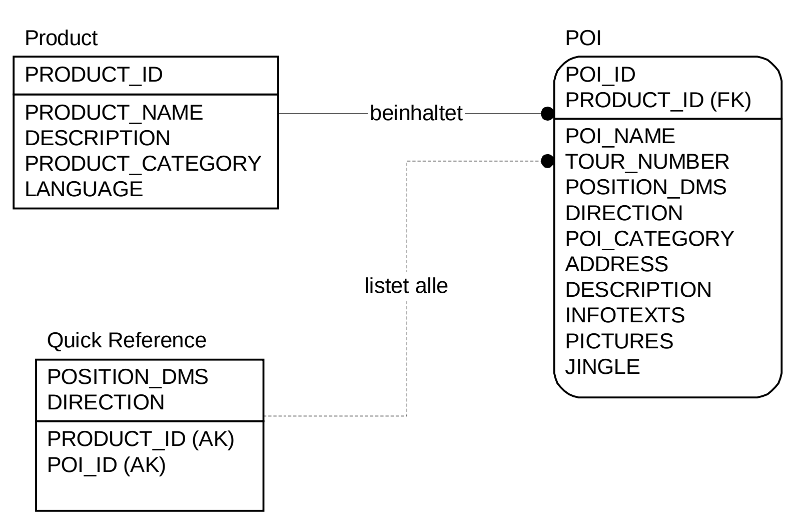


Abbildung 3.12: Entity-Relationship-Diagramm des Datenbankentwurfs

3.2.6.1 POI-Tabelle

In der POI-Tabelle wird genau ein POI detailliert beschrieben. Über den Fremdschlüssel PRODUCT_ID muß **immer** eine Beziehung zu einem Produkt hergestellt werden. Ansonsten ist der POI nicht eindeutig identifizierbar und somit unbrauchbar. Die Ausgabe/Anzeige von POI Inhalten kann durch verschiedene Instanzen (TTS, HTML-Browser, etc.) erfolgen.

Attributname	Art	Beschreibung	Größe
POI_ID	Zahl	Primary Key der POI-Tabelle. Eindeutige Identifizierung eines jeden POI's innerhalb eines Produktes. Erst durch die Verbindung mit der PRODUCT_ID wird der POI innerhalb des Systems eindeutig gekennzeichnet.	2 Byte
PRODUKT_ID	Zahl	Foreign Key des POI's. Stellt die Beziehung zu einem bestimmten Produkt her. Nur unter Angabe der PRODUKT_ID kann ein POI, mit seiner ID, eindeutig identifiziert werden.	2 Byte
POI_NAME	Text	Name des POI. Möglicherweise keine Ausgabe per TTS, aber Anzeige in der MMI.	Max. 80 Zeichen
TOUR_NUMBER	Zahl	Falls der POI Teil eines Produktes der Kategorie TOUR ist, wird diese Nummer benötigt um die Reihenfolge der abzufahrenden Punkte festzulegen. Ansonsten kann dieses Feld leer bleiben.	2 Byte
POSITION_DMS	Struktur	Möglichst genaue geographische Längen- und Breitenangabe des POIs.	Größe der Struktur
DIRECTION	Gleitzahl	Winkel der Bewegungsrichtung des POIs. Aktionen können erfolgen, wenn das Fahrzeug diese Richtung bewegt während die Position des POI durchfahren wird.	Größe der Gleitzahl
POI_CATEGORY	Enum	Einteilung der POI's in Kategorien. Jeder POI kann nur einer Kategorie zugeordnet werden. Diese ist allerdings unabhängig von der PRODUCT_CATEGORY. Gültige POI Kategorien sind derzeit: RADAR_TRAP (Radarfalle), RADAR_TRAFFICLIGHT (Blitzampel), SIGN(Hinweisschild), SIGHTSEEING (Sehenswürdigkeit).	Max. Anzahl von POI_CATEGORY
ADDRESS	Struktur	Enthält Ortsangaben in Klartext. Struktur mit Land, Bundesstaat, Kreis, Postleitzahl, Ort, Ortsteil, Straße. Alle Einträge sind optional, da diese Angaben stark vom jeweiligen Kulturkreis und der Bedeutung des POI abhängig sind.	Größe der Struktur
DESCRIPTION	Text	Kurzbeschreibung des POI. (Ausgabe per TTS denkbar)	Max. 80 Zeichen
INFOTEXTS	Link	Verweis auf eine Datei, die weitere Informationen zu einem POI enthält. Diese wird dem Reisenden Per TTS vorgelesen.	Max. 80 Zeichen für Dateinamen, der mit .info enden muß
PICTURES	Pfad	Falls Bilder zu einem POI existieren wird hier eine Pfadangaben gemacht	Pfadname
JINGLE	Enum	Bestimmt die Art des Jingles, welches vor der Sprachausgabe zu spielen ist. Eins von: NO_JINGLE (kein Jingle), INFOTEXT, ADVICE (Hinweis), WARNING.	Max. Anzahl von JINGLE

Tabelle 3.14: Die POI-Tabelle

3.2.6.2 Product-Tabelle

Die Product-Tabelle dient zum Bündeln von POI's. Ein Produkt kann eins bis n POI's beinhalten. Die Relation zwischen POI- und Product-Tabelle wird über das Attribut PRODUCT_ID hergestellt. Dieses ist im Table POI ein Fremdschlüssel.

Attributname	Art	Beschreibung	Größe
PRODUCT_ID	Zahl	Primary Key der Tabelle Product. Eindeutige Identifizierung eines jeden Produktes, das der Benutzer auf sein System laden kann. Die Vergabe von PRODUCT_ID's müßte auf dem Backbone stattfinden.	2 Byte
PRODUCT_NAME	Text	Enthält die, möglicherweise kommerzielle, Bezeichnung des Produktes. Kann zur Anzeige in der MMI genutzt werden. Der Name eines jeden Produktes soll eindeutig sein.	Max. 80 Zeichen
DESCRIPTION	Link	Verweis auf eine HTML-Datei, die dem Benutzer angezeigt werden kann, um den Inhalt des gewählten Produktes zu vermitteln.	Max. 80 Zeichen für Dateinamen, der mit .html enden muß
PRODUCT_CATEGORY	Enum	Produkte werden in Kategorien eingeteilt. Über diese Einteilung können Produkte verschieden behandelt werden. Die Kategorien sind: TOUR, COUNTRY, STATE, REGION, CITY und INFO.	Max. Anzahl von PRODUCT_CATEGORY
LANGUAGE	Enum	Gibt an für welchen Sprachraum ein Produkt geeignet ist. Erzwingt aber nicht das alle enthaltenen POI's in dieser Sprache sind. Vorgesehen sind Kürzel für D (Deutschland), DK (Dänemark), GB (Großbritannien), F (Frankreich), I (Italien), NL (Niederlande) und E (Spanien).	Max. Anzahl von LANGUGAGE

Tabelle 3.15: Die Product-Tabelle

3.2.6.3 Quick Reference-Tabelle

Die Quick Reference-Tabelle soll die Effizienz im laufenden Betrieb steigern. Bei aktiver Reiseinfo werden ständig die Positionen aller POI's mir der aktuellen Position verglichen. Wird zur aktuellen Position ein passender POI gefunden, so kann sein Inhalt über PRODUCT_ID und POI_ID referenziert werden.

Attributname	Art	Beschreibung	Größe
POSITION_DMS	Struktur	Möglichst genaue geographische Längen- und Breitenangabe des POIs.	Größe der Struktur
DIRECTION	Gleitzahl	Winkel der Bewegungsrichtung des POIs. Aktionen können erfolgen, wenn das Fahrzeug diese Richtung bewegt während die Position des POI durchfahren wird.	Größe der Gleitzahl
PRODUCT_ID	Zahl	Primary Key der Produkt-Tabelle. Eindeutige Identifizierung eines jeden Produktes, das der Benutzer auf sein System laden kann	2 Byte
POI_ID	Zahl	Primary Key der POI-Tabelle. Eindeutige Identifizierung eines jeden POI's innerhalb eines Produktes.	2 Byte

Tabelle 3.16: Die Quick Reference-Tabelle

3.3 Lösung der Aufgaben

Um die Aufgabenstellung zu lösen, mußten verschiedene Punkte bearbeitet werden. Ein Großteil der Lösungen und Ansätze ist bereits in den vorhergehenden Kapiteln beschrieben. Hier sollen aber noch einmal Lösungen und Lösungswege zu ganz konkreten Problemen aufgezeigt werden. Ein graues Kästchen, rechts unterhalb der Kapitelübeschriften, markiert ob eine Implementierung vorhanden ist.

3.3.1 Auffinden einer geeigneten GPS-Quelle

Implementierung in der Klasse „GPS“

Durch die Systemkonfiguration gibt es weder eine Vorschrift, welche GPS-Quelle ein Benutzer anschließen muß, noch über welche der beiden seriellen Schnittstellen er dies tun sollte. Somit muß zur Laufzeit ermittelt werden auf welche Art eine GPS-Quelle tatsächlich angeschlossen ist.

Zur Lösung dieses Problems könnte man den Benutzer eine Menü-Gesteuerte Konfiguration durchführen lassen. Um den Reiseführer aber komfortabler zu gestalten, ist eine Funktionalität vorgesehen, welche die seriellen Schnittstellen[37] systematisch nach GPS-Quellen absucht. Ein Bedienereingriff ist nicht notwendig.

Begonnen wird mit der Suche nach einem angeschlossenen Navigationssystem. Zunächst an der ersten seriellen Schnittstelle (COM1) und dann an der zweiten seriellen Schnittstelle (COM2), falls die Suche an COM1 erfolglos blieb. Konnte keine Verbindung zu einem Navigationssystem hergestellt werden, so wird nach der Garmin Mouse gesucht. Zu Beginn auch wieder an der ersten und im Falle eines Mißerfolges an der zweiten seriellen Schnittstelle.

Das Ergebnis der Suche wird wie folgt bewertet:

- Wurde ein Navigationssystem gefunden können sämtliche Funktionen des Reiseführers angeboten werden.
- Ist nur die Garmin Mouse, als GPS-Quelle, verfügbar steht dem Reisenden mindestens das Leistungsmerkmal Reiseinfo zur Verfügung. Die Tourenführung ist ohne Navigationssystem nicht möglich.
- Konnte keine GPS-Quelle gefunden werden, so sind nur Funktionen verfügbar, die prinzipiell ohne GPS arbeiten. Also, das „Reiseführer browsen“ und das „Bearbeiten der Reisedatenbank“.

Bereits bei der Suche nach einer GPS-Quelle wurden die entsprechenden Kommunikationswege initialisiert. Es muß lediglich dafür gesorgt werden das anderen Programmteilen der Zugriff darauf ermöglicht wird. Änderungen, die zur Laufzeit am Anschluß der GPS-Quelle vorgenommen werden, können allerdings nicht berücksichtigt werden.

3.3.1.1 Initialisierung der GPS Mouse

Es wird zunächst der frei erhältliche Treiber „gpsd“[38] gestartet. Dieser Treiber initialisiert und überwacht die serielle Schnittstelle. Er übersetzt die Positionsdaten eines GPS-Empfängers in ein einfaches Format und stellt sie dann auf einem BSD Socket[39] zur Verfügung. Dieser Socket wird konnektiert und kann von anderen Systemteilen angesprochen werden.

3.3.1.2 Initialisierung des Navigationssystems

Nach der Interface Spezifikation [F3] wird die serielle Schnittstelle auf 9600 Bits pro Sekunde, gerade Parität, acht Datenbits und ein Stopbit eingestellt. Auf die serielle Schnittstelle wird dann ein Telegramm zur Initialisierung geschrieben. Als Antwort wird ein Telegramm mit positiver Bestätigung erwartet und die Initialisierung ist abgeschlossen. Inhalte der Telegramme sind hier nicht wesentlich und können daher in [F3] nachgelesen werden.

---

[37] Programmieren der seriellen Schnittstelle unter Zuhilfenahme von [B6] Kapitel 20 und [B7] S. 275.

[38] Unter anderem frei zum Download bei freshmeat.net [W22]

[39] Ähnlich wie bei Festival in Kapitel 2.4.2

3.3.2 Kommunikation mit der GPS Mouse

Implementierung in der Klasse „Garmin“

Wie eingangs erwähnt arbeitet die GPS Mouse nach dem NMEA Protokoll. Es werden kommaseparierte ASCII-Sätze übertragen, die vom Reiseführer ausgewertet werden müssen.

Mit einer Aktualisierung von einmal pro Sekunde sendet die GPS Mouse kontinuierlich verschiedene Sätze aus. Für den Reiseführer wird zunächst nur der bereits gezeigte Satz $GPRMC benötigt. Die übertragenen NMEA-Sätze werden in einen Puffer eingelesen und nach der Zeichenfolge $GPRMC gefiltert. Zur Erinnerung noch einmal der Aufbau des Satzes in Abbildung 3.13.

$GPRMC,<1>, <2>,<3>, <4>,<5>,<6>,<7>,<8>,<9>,<10>,<11>*hh<CR><LF>

Abbildung 3.13: Aufbau des Satzes $GPRMC

Die Zeichen von $ bis <LF> werden von einer weiteren Funktion analysiert. Hierbei werden alle Inhalte die auf ein Komma folgen in ein neues Feld geschrieben. Inhalte der Felder werden dann je nach Bedarf in Ganzzahlen, Gleitzahlen oder ASCII-Zeichen umgesetzt, die wiederum in einer neuen Struktur untergebracht werden.

3.3.2.1 Bedeutung der $GPRMC Datenfelder

Aufbau und Bedeutung des Satzes $GPRMC sind ein Auszug aus [S4].

<1> Aktuelle UTC-Zeit[40] im Format hhmmss

<2> Status, A = gültige Position, V = Empfänger Warnung

<3> Latitude im Format ddmm.mmmm (führende Nullen werden mit Übertragen)

<4> Hemisphäre[41] der Latitude, N oder S

<5> Longitude im Format dddmm.mmmm (führende Nullen werden mit Übertragen)

<6> Hemisphäre der Longitude, E oder W

<7> Geschwindigkeit in Knoten von 0.0 bis 999.9

<8> Bewegungsrichtung in Grad von 000.0 bis 359.9 (führende Nullen werden mit Übertragen)

<9> Aktuelles UTC-Datum im Format ddmmyy

<10> Magnetische Abweichung in Grad von 000.0 bis 180.0 (führende Nullen werden mit Übertragen)

<11> Richtung der Magnetischen Abweichung, E oder W (westliche Abweichung addiert sich auf die Bewegungsrichtung)

3.3.2.2 Umsetzung der Datenfelder

Im Folgenden soll erläutert werden, wie die einzelnen Felder des NMEA-Satzes behandelt werden:

<1> Die UTC-Zeit wird in eine Struktur übersetzt, die Stunde, Minute und Sekunde enthält. Um den Zeitunterschied zwischen UTC-Zeit und Mitteleuropäischer Zeit (MEZ) auszugleichen wird, je nach Sommer- oder Winterzeit, ein bestimmter Wert zur aktuellen Stunde dazu addiert. Einstellungen der Zeitzone durch den Benutzer sind nicht vorgesehen.

<2> Das Feld wird nach „A“ oder „V“ ausgewertet und speichert das Ergebnis in einer Variablen, die anzeigt ob eine gültige Position empfangen wurde, oder nicht.

---

[40] Universal Time Coordinated ist die Standard Zeit, die für jeden Ort auf der Erde gebräuchlich ist. Früher bekannt und oft noch bezeichnet als mittlere Greenwich-Zeit (GMT) oder Weltzeit.

[41] Halbkugel

<3> Umsetzung des Breitengrades in die Struktur[42]. Hierbei werden die Werte für den Winkel (dd) und die Minuten (mm) direkt übernommen. Der Wert für die Gleitzahl des Sekundenanteils muß allerdings erst berechnet werden. Die vier Zeichen, die den Nachkommateil der Minuten angeben, werden in eine Ganzzahl übersetzt, durch den Faktor 10.000 geteilt, um eine Gleitzahl zu erhalten und mit 60 multipliziert, um auf Sekunden zu kommen.

Hierzu ein Beispiel:

- Wert von <3> ddmm.mmmm = 5034.8723
- dd = 50 Grad
- mm = 34 Minuten
- .mmmm = 8723
- 8723 geteilt durch 10.000 = 0,8723
- 0,8723 multipliziert mit 60 = 52,338 Sekunden
- Gesamtausdruck: 50° 34‘ 52,338“

Tabelle 3.17: Beispiel für die Umsetzung von Koordinaten

<4> Gibt an ob der Wert für den Breitengrad nördlich oder südlich des Äquators gültig ist. Im System wird nördlich als GEO_NORTH mit 0 und südlich als GEO_SOUTH mit 1 definiert.

<5> Umsetzung des Längengrades in die Struktur dms_s. Ähnlich wie beim Breitengrad werden die Werte für Winkel (ddd) und Minuten (mm) direkt übernommen. Der Wert für die Gleitzahl des Sekundenanteils muß wie im obigen Beispiel berechnet werden.

<6> Zeigt an ob der Wert für den Längengrad östlich oder westlich von Greenwich gültig ist. Im System wird östlich als GEO_EAST mit 0 und westlich als GEO_WEST mit 1 definiert.

<7> Die Geschwindigkeit, angegeben in Knoten, wird umgerechnet in Kilometer pro Stunde. Hierzu wird der gelesene Wert mit 1,825 multipliziert, um km/h zu erhalten. Ein Knoten entspricht einer Seemeile pro Stunde, das ist gleich 1,825 km/h.

<8> Eine Angabe der Bewegungsrichtung wird Relational zum Nordpol gemacht. Bewegt man sich genau in nördlicher Richtung so ist der entsprechende Winkel 0°. Alle Abweichungen davon werden im Uhrzeigersinn angegeben. Siehe hierzu die folgende Abbildung 3.14.

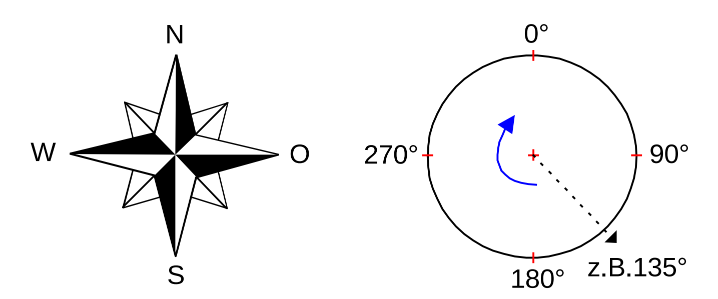


Abbildung 3.14: Messen der Bewegungsrichtung

<9> Ähnlich wie die Uhrzeit wird auch das Datum im UTC-Format übertragen. Es kann einfach in Tage, Monat und Jahr übersetzt werden. Wobei das Jahr ein Wert bezüglich des aktuellen Jahrhunderts darstellt. Für 2002 wird demnach eine „02“ übertragen. Gegenwärtig spielt das Datum eine untergeordnete Rolle für den Reiseführer.

<10> Der übertragene Wert gibt die Abweichung zwischen „wahrem“ Norden[43] und magnetischem Norden[44] an. Die scheinbare Lage des magnetischen Nordens ist abhängig von der Position, insbesondere des Breitengrades, auf der Erde. Kenntnis der Magnetischen Abweichung ist notwendig um die exakte Bewegungsrichtung <8> bestimmen zu können.

---

[42] dms_s steht für „Degrees, Minutes, Seconds“ und ist beschrieben in Kapitel 3.3.5

[43] Geographisch nördlichster Punkt der Erde.

[44] Magnetisches Nordende des „Stabmagnets“ der Erde.

<11> Bei westlicher Abweichung wird der Wert aus <10> zur Bewegungsrichtung <8> dazu addiert. Ist die Abweichung östlich, so wird der Wert abgezogen.[45]

Versuche zur Anbindung der GPS Mouse wurden mit dem selbst entwickelten Testprogramm „garmin_serial.c“ durchgeführt.

3.3.3 Kommunikation mit dem Navigationssystem

Implementierung in der Klasse „Nav“

Die Anbindung des Navigationssystems geschieht, wie eingangs erwähnt, über das proprietäre Protokoll „CCI“. Teilnehmer am CCI Protokoll sind das Navigationssystem und alle externen Geräte, die als CCI Gerät bezeichnet werden. Eine Initialisierung des Navigationssystems ist bereits beim auswählen der GPS-Quellen geschehen.

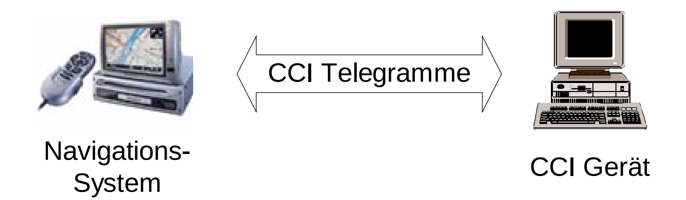


Abbildung 3.15: Verbindung von Navigation und externem Gerät

Das Protokoll basiert auf der Übertragung festgelegter Zeichenketten, die als Telegramme bezeichnet werden. Aufbau der Telegramme kann bei Bedarf den Siemens VDO Spezifikationen [F3] und [F4] entnommen werden. Trotz vorhergehender Initialisierung sollte noch der Telegrammindex zurückgesetzt werden, bevor Telegramme versendet werden. Zusätzlich kann die Arbeitsweise des Navigationssystems, per Telegramm, konfiguriert werden. Für diese Arbeit wird als Konfiguration folgendes festgelegt:

- Die Zeitdauer in der ein Telegramm bestätigt werden muß, bevor es erneut gesendet wird, beträgt eine Sekunde.
- Für den Fall einer Zeitüberschreitung wird das nicht bestätigte Telegramm null mal wiederholt.
- Periodisch gesendete Telegramme werden einmal pro Sekunde aktualisiert.
- Das Navigationssystem wartet nicht auf die Bestätigung eines Telegramms, wenn erneut ein Telegramm zum selben Empfänger gesendet wird.

Maßgeblich für diese Arbeit ist das Telegramm, zum Übertragen der aktuellen Fahrzeugposition und das zum Setzen neuer Zielkoordinaten für die Navigation. Aufbau und Umsetzung dieser Telegramme wird in den folgenden Kapiteln behandelt.

3.3.3.1 Übertragen der aktuellen Fahrzeugposition

Bevor das Navigationssystem mit der Übertragung aktueller Positionsdaten beginnt, muß es zunächst dazu aufgefordert werden. Dies geschieht mit Hilfe des in Abbildung 3.16 beschriebenen Telegramms. Die Begriffe „CCI Overhead“ und „Checksumme“ werden hier nicht näher erläutert, können aber in [F3] nachgelesen werden.

CCI Overhead	Function Code	DB0	Check-summe

Abbildung 3.16: Telegramm zum anfordern von Positionsdaten

<Fct. Code> Funktionscode, der das Navigationssystem zum Senden der aktuellen Positionsdaten auffordert. Der hier benötigte Wert ist 0x60 Hex.

[45] Ausführliches zu diesem Thema ist bei den „Scouting Resources“ [W21] zu finden.

<DB0> Bytewert dessen Bits die folgende Bedeutung übernehmen:

Bit 0: 0 => Anforderung der Position als Koordinaten

1 => Anforderung der Position als postalische Adresse

Bit 1-3: Reserviert für zukünftige Verwendungen

Bit 4: 1 => Aufforderung zum starten der periodischen Aktualisierung

0 wird nicht verwendet

Bit 5: 1 => Aufforderung zum stoppen der periodischen Aktualisierung

0 wird nicht verwendet

Bit 6,7: Reserviert für zukünftige Verwendungen

Für diese Arbeit wird die Position in Koordinaten benötigt. Für <DB0> wird mit Bitoperationen ein Wert zusammengesetzt, der 0x10 Hex entspricht.

Wie in der Bedeutung von Bit 4 und Bit 5 gesehen, kann das angegebene Telegramm zum starten und stoppen der periodischen Aktualisierung verwendet werden. Hat die periodische Aktualisierung begonnen so wird, einmal pro Sekunde, das in Abbildung 3.17 gezeigte Telegramm vom Navigationssystem gesendet.

CCI Overhead	Function Code	DB0	...	DB17	Check-summe

Abbildung 3.17: Telegramm das die Positionsdaten enthält

Eine genaue Beschreibung der Datenfelder <DB0> bis <DB17> ist im folgenden Text gegeben. Zur besseren Übersicht ist eine entsprechende Tabelle im Anhang B zu finden.

<DB0> Besitzt immer den Wert 0x01h als Bestätigung, dass die Position als Koordinaten übertragen wird.

<DB1> Enthält sämtliche Statusinformationen zum GPS Empfang. Mit Hilfe von Bitoperationen werden die einzelnen Bits ihrer Bedeutung nach ausgewertet und dem Reiseführer verfügbar gemacht.

<DB2> und <DB3>

Als Referenz wird hier auch UTC-Zeit verwendet. Zeitzonen können durch den Benutzer im Navigationssystem eingestellt werden. Die übertragenen Werte werden vom Reiseführer direkt übernommen.

<DB4> bis <DB6>

Der Wert für das Jahr muß mittels Offset berechnet werden. Ansonsten übernimmt der Reiseführer das Datum so wie übertragen.

<DB7> bis <DB14>

Die Werte für Längen- und Breitengrad werden in die Struktur dms_s umgesetzt. Angaben zu den Hemisphären werden, wie im System definiert, übernommen.

<DB15> und <DB16>

Um die Gradzahl der Fahrtrichtung zu erhalten muß der hier empfangene Wert durch zehn geteilt werden. Dabei ist <DB15> das höherwertige und <DB16> das niederwertige Byte.

<DB17> Gibt die aktuelle Geschwindigkeit des Fahrzeugs an. Der übertragene Wert ist ganzzahlig und wird in eine Gleitzahl konvertiert. Beim Rückwärtsfahren ist der Wert stets null.

3.3.3.2 Setzen eines neuen Ziels

Um dem Leistungsmerkmal Tourenführung gerecht zu werden ist es nötig ein neues Ziel auf dem Navigationssystem zu setzen. Nach Senden des folgenden Telegramms, an das Navigationssystem, wird die Zielführung zur übertragenen Position gestartet.

CCI Overhead	Function Code	DB0	...	DB12	Check-summe

Abbildung 3.18: Senden einer Position für die Zielführung

<Fct. Code> Funktionscode, der anzeigt das Positionsdaten für die Zielführung gesendet werden. Der hier benötigte Wert ist 0x6C Hex.

<DB0> 0x01 Hex => Position als Koordinaten

<DB1> 0x01 Hex => Ziel für Navigationssystem

<DB2> bis <DB5> Winkelangaben und Hemisphäre für den Breitengrad, wie in Tabelle B.1

<DB6> bis <DB9> Winkelangaben und Hemisphäre für den Längengrad, wie in Tabelle B.1

<DB10> bis <DB12> Reserviert für zukünftige Verwendungen

Versuche zur Anbindung des Navigationssystems wurden mit dem selbst entwickelten Testprogramm „nav_serial.c“ durchgeführt.

<u>Hinweis:</u>

Für den generellen Testbetrieb ist es sehr hilfreich, dass man das Navigationssystem im sogenannten „Demomode“ betreiben kann. Hierbei wird eine navigierte Fahrt zwischen zwei frei wählbaren Orten simuliert. Während der gesamten Simulationsfahrt gibt das Navigationssystem Positionsdaten aus, die von der digitalen Karten-CD stammen.

3.3.4 Integration von Festival

3.3.4.1 Installation

Zur Installation des Sprachsynthese Systems Festival, der Universität Edinburgh, müssen verschiedene Schritte durchgeführt werden. Zunächst sind Dateiarchive von der Internet-Ressource[46], der Universität herunterzuladen. Es handelt sich hierbei um die Distribution von Quellcode, der zu einem lauffähigen Programm kompiliert werden muß.

Bevor allerdings mit der Installation von Festival begonnen werden kann müssen die sogenannten „Speech Tools“ installiert werden, die ebenfalls als Quellcode verfügbar sind. Nach dem Kompilieren von Speech Tools stehen verschiedene Binaries[47] und Bibliotheken zur Verfügung. Diese werden von Festival, nur zur Kompilierungszeit, benötigt.

Nach dem Kompilieren der Festival Distribution stehen bereits lauffähige Programme zur Verfügung und erste Sprachausgaben können gemacht werden. In der Standarddistribution beherrscht Festival nur die Englische Sprache. Um jedoch Deutsch verwenden zu können sind weitere Schritte notwendig.

Eine detaillierte Beschreibung der benötigten Dateien, schrittweises vorgehen und somit erfolgreiche Installation von Festival ist im Anhang C beschrieben.

---

[46] Siehe hierzu [W15]

[47] Zu einem lauffähigen Programm kompilierter Quellcode

Das im Anhang beschriebene vorgehen kann und sollte auf einem vollwertigen Linux-System durchgeführt werden. Um auf dem Zielsystem (MC5400) den Speicherbedarf so gering wie möglich zu halten werden nur die absolut notwendigen Dateien dorthin kopiert. Es ergibt sich eine Struktur wie in Abbildung 3.19 zu sehen.

Flash 1 [64MB]

[7,7 MB] kernel	[41,8 MB, 4,5 MB frei] root	[14,5 MB, 3,5 MB frei] user
	Davon Festival Binaries und est Libs ca. 9,8 MB	Davon MBROLA und Deutsch ca. 10MB

Flash 2 [32 MB]

[32 MB, 6MB frei]
Davon benötigte Dateien aus festival/lib ca. 26 MB

Abbildung 3.19: Festival auf dem Zielsystem

Die von Festival benötigten Dateien werden auf zwei Flash-Karten aufgeteilt. Bei Verwendung einer größeren Flash 1, z.B. 128 MB, wäre diese Aufteilung nicht notwendig[48]. Benötigte Dateien der zweiten Flash werden zur Laufzeit im System gemountet.

3.3.4.2 Verwendete API

Implementierung in der Klasse „Speech“

In der Vorbetrachtung (Kapitel 2) des Text to Speech Systems werden die verschiedenen Schnittstellen zu Festival angesprochen. Die in dieser Arbeit verwendete Schnittstelle ist die Server/Client API.

Ein Systemaufruf erzeugt einen neuen Prozess und startet Festival im Servermodus. Auf dem BSD Socket Port Nummer 1314 wird die Kommunikation zwischen Server und Client durchgeführt. Mittels einer Initialisierungsfunktion wird der Reiseführer als Client mit dem Sprachserver verbunden. Darauf folgt die Einstellung der gewünschten Sprache. Es kann Deutsch oder Englisch gewählt werden. Für Testzwecke und Demonstrationen wurde häufig Deutsch verwendet.

Um Sprachausgaben machen zu können, steht eine Funktion zur Verfügung; speech_out_speek(). Diese Funktion erhält den auszugebenden Text, modifiziert diesen und schreibt auf den geöffneten Port. Auf Serverseite wird dieser Text durch den Scheme-Interpreter ausgewertet. Der Interpreter reagiert auf den übertragenen Text so, als wäre er auf der Kommandozeile eingegeben worden.

Hierzu ein kleines Beispiel:

- void Speech::**speech_out_speek**(char* pText)
- wird aufgerufen mit dem pText: **„hallo welt“**
- Die Funktion bildet daraus den String: **„(SayText “hallo welt”)”**
- Exakt so, mit Klammern, wird dieser an den Interpreter übertragen.
- Der wendet dann die Funktion SayText auf „hallo welt“ an und es kommt zur Sprachausgabe.

Tabelle 3.18: Beispiel zur Sprachausgabe-Funktion

[48] 32MB und 64MB Karten standen für diese Arbeit zur Verfügung. Bei Karten mit 128MB waren die Anschaffungskosten zu hoch.

3.3.4.3 Laufzeitprobleme

Nach Aufruf der Sprachsynthese tritt eine Verzögerung bis zur tatsächlichen akustischen Ausgabe auf. Bei der Benutzung von Festival konnte festgestellt werden, dass die Zeitdauer unterschiedlich groß ist. Je nach Länge des Textes und verwendeter Hardware kann es zu deutlichen Unterschieden kommen. Ähnlich wie bei den Abweichungen von GPS handelt es sich hier um subjektive Eindrücke, die durch eine Meßreihe belegt werden sollen.

Zum Versuchsaufbau werden verschiedene Computersysteme und Beispielsätze verwendet. Bei den Computersystemen handelt es sich um einen handelsüblichen Desktop PC (Intel Pentium III 750 MHz, mit 256MB Hauptspeicher), einen Laptop (Intel Pentium II 300 MHz, mit 96MB Hauptspeicher) und zwei MC5400. Eines der beiden MC5400 ist ein sogenanntes „B-Muster“ mit 64MB Hauptspeicher und das andere ist ein „C-Muster“ mit 32MB. Die Beispielsätze sind in Tabelle 3.19 dargestellt.

Satz Nr.	Auszugebende Beispielsätze	Anzahl Zeichen
1	„Hallo Welt!“	10
2	„Als innovativer Partner fuer Automotive Information und Kommunikation unterstuetzen wir den Fahrer.“	90
3	„Weithin sichtbares Wahrzeichen Wetzlars ist der Dom. Die heute uebliche Bezeichnung tritt schon im 17. Jahrhundert auf, hat sich aber wohl erst im 18.Jahrhundert eingebuergert. Das monumentale, gleichwohl bisher unvollendete Bauwerk bildet den architektonischen Mittelpunkt der Altstadt.“	251

Tabelle 3.19: Beispielsätze für den Zeitvergleich

Zum Erfassen der Zeitdauer, für die Synthetisierung eines jeden Satzes, werden Debug-Werkzeuge verwendet, die nicht zur Standarddistribution von Festival gehören. Es gibt die Möglichkeit einen Satz beliebig oft, in einer Schleife, zu synthetisieren und die jeweiligen Ergebnisse in einer Textdatei abzulegen. Abbildung 3.20 zeigt einen Auszug aus der Ergebnisdatei „dom.txt“ bezüglich des Beispielsatzes Satzes Nummer drei, synthetisiert auf dem Desktop PC. Genau 13 Werte werden für jede Synthetisierung hintereinander in die Datei geschrieben. Die Bedeutung der einzelnen Werte wird hier nicht besprochen.

```
Initialize took 0.103 ms
Text took 1.007 ms
Token_POS took 2.308 ms
Token took 165.532 ms
POS took 0.140 ms
Phrasify took 5.603 ms
Word took 311.233 ms
Pauses took 6.001 ms
Intonation took 14.002 ms
PostLex took 22.606 ms
Duration took 28.132 ms
Int_Targets took 118.708 ms
Wave_Synth took 584.619 ms
Initialize took 0.994 ms
Text took 1.047 ms
Token_POS took 2.073 ms
Token took 160.619 ms
...
```

Abbildung 3.20: Auszug der Ergebnisdatei „dom.txt“

In einem nächsten Schritt können die vorher erfassten Daten automatisch analysiert werden. Die 13 Werte eines jeden Synthetisierungs-Vorgangs werden zusammen addiert und ergeben eine Gesamtzeit. Um die Genauigkeit der Ergebnisse zu erhöhen wird jeder Satz 100 mal synthetisiert. Aus den 100 Vorgängen werden dann Mittelwerte für alle Gesamtzeiten berechnet. Es wird der arithmetische Mittelwert (Mean) und der Durchschnittswert (Median) angegeben. In Abbildung 3.21 wird eine solche Analyse analog zum vorher verwendeten Satz drei dargestellt. Auch hier wird nicht weiter auf die einzelnen Werte eingegangen.

dom.txt → dom_re.txt

Number of data sets 100

Lines per data set 13

Initialize	1.025 ms	0.19 %
Text	1.065 ms	0.20 %
Token_POS	2.154 ms	0.40 %
Token	173.978 ms	32.12 %
POS	0.149 ms	0.03 %
Phrasify	5.683 ms	1.05 %
Word	51.714 ms	9.55 %
Pauses	2.935 ms	0.54 %
Intonation	14.026 ms	2.59 %
PostLex	22.351 ms	4.13 %
Duration	28.887 ms	5.33 %
Int_Targets	120.676 ms	22.28 %
Wave_Synth	117.014 ms	21.60 %

541.657 ms (Mean)
533.663 ms (Median)

Abbildung 3.21: Analyse der Ergebnisdatei in „dom_re.txt“

Abschließend ist in Tabelle 3.20 das analysierte Ergebnis aller Synthetisierungen, auf den verschiedenen Systemen, abgebildet. Hierbei lassen sich leicht die eingangs angesprochenen Unterschiede ablesen. Deutlich wird, dass Festival nicht für den Einsatz auf dem MC5400 geeignet ist. Die Zeiten für Synthetisierung, eines längeren Satzes, liegen bei circa 20 Sekunden. Für diese Arbeit und zu Testzwecken wird Festival aber dennoch als ausreichend angesehen.

Satz Nr.	Schnitt	Desktop PC	Laptop	MC5400 64MB	MC5400 32MB
1	Mean	45,689 ms	135,827 ms	1143,673 ms	1106,274 ms
	Median	45,271 ms	117,876 ms	1134,101 ms	1087,918 ms
2	Mean	162,925 ms	470,280 ms	5647,628 ms	5380,573 ms
	Median	145,418 ms	449,088 ms	5615,099 ms	5330,029 ms
3	Mean	541,657 ms	1725,337 ms	20834,262 ms	19989,773 ms
	Median	533,663 ms	1696,559 ms	20797,298 ms	19926.588 ms

Tabelle 3.20: Zeitvergleich zur Sprachausgabe

3.3.5 Geographische Werkzeuge

Implementierung in der Klasse „Geo“

Zu den Geographischen Werkzeugen gehören Datentypen und Algorithmen. Diese sind notwendig um von GPS-Quellen gelieferte Daten zu Konvertieren und zu Speichern. Ausserdem können einfachste Berechnungen, z.B. der Entfernung durchgeführt werden.

3.3.5.1 Datentypen

Wie in vorhergehenden Kapiteln beobachtet, liefern Navigationssystem und GPS Mouse Positionsdaten in verschiedenen Formaten und mit verschiedenen Details. Da der Reiseführer generell unabhängig, von der gewählten GPS-Quelle, sein soll werden allgemeine Datentypen festgelegt. In diese Datentypen hinein müssen dann die Werte der GPS-Quellen konvertiert werden.

Name:	date_time_s
Beschreibung:	Speichert Datum- und Zeitangaben. Eine optionale Verwendung der einzelnen Strukturteile ist möglich. Für die Darstellung des Kalenderjahres wird hier kein Offset verwendet, z.B. Das Jahr 2002 wird auch als Ganzzahl 2002 abgelegt.
Struktur:	short Hour;
	short Minute;
	short Second;
	short Day;
	short Month;
	unsigned short Year;

Tabelle 3.21: Datentyp date_time_s

Name:	dms_s	
Beschreibung:	DMS steht als Abkürzung für „Degrees, Minutes und Seconds“. Es handelt sich um die Angabe eines Winkels im Format: Grad° Minuten‘ Sekunden, Zehntelsekunden“ Der Datentyp dms_s wird verwendet um Angaben für Breiten- und Längengrad abzubilden. Die Werte für Degrees und Minutes sind ganzzahlig, während durch die Gleitzahl für Seconds eine beliebige Genauigkeit erreicht werden kann.	
Struktur:	unsigned char	Degress;
	unsigned char	Minutes;
	double	Seconds;

Tabelle 3.22: Datentyp dms_s

Name:	position_dms
Beschreibung:	Im Datentyp position_dms werden verschiedene Attribute zusammengefaßt, um eine genaue Position (Längen- und Breitengrad) auf der Erde abzubilden. Zusätzlich wird eine Information über die aktuelle Bewegungsrichtung angegeben. Die Werte für Längen- und Breitengrad sind hier als dms_s abgelegt. Typisch für die GPS-Quellen ist, dass Positionsdaten in einzelnen Feldern übergeben werden. Hier eignet sich dieser Datentyps besonders, da zunächst nur wenige Konvertierungen durchgeführt werden müssen.
Struktur:	dms_s Latitude;
	dms_s Longitude;
	unsigned char LatHemisphere;
	unsigned char LongHemisphere;
	double Direction;

Tabelle 3.23: Datentyp position_dms

Name:	position_dbl
Beschreibung:	Im Datentyp position_dbl werden verschiedene Attribute zusammengefaßt, um eine genaue Position (Längen- und Breitengrad) auf der Erde abzubilden. Zusätzlich wird eine Information über die aktuelle Bewegungsrichtung angegeben. Die Werte für Längen- und Breitengrad sind hier als Gleitzahl abgelegt. Die Verwendung diesen Datentyps eignet sich besonders für Entfernungsberechnungen, da die entsprechenden Formeln dazu Gleitzahlen verlangen.
Struktur:	double Latitude; double Longitude; unsigned char LatHemisphere; unsigned char LongHemisphere; double Direction;

Tabelle 3.24: Datentyp position_dbl

3.3.5.2 Algorithmen - Sphärische Trigonometrie

Grundlage für Algorithmen des Reiseführers ist die Entfernungsberechnung zwischen zwei Punkten, auf der Erde. Mit dem Wissen über die Distanz (Luftlinie) zu einem Objekt kann z.B. entschieden werden ob es zu einer Ausgabe kommt, oder nicht. Um den Abstand zweier Punkte, gegeben in der Form position_dms, berechnen zu können müssen drei Schritte ausgeführt werden:

1. Konvertierung der Winkelwerte von position_dms in Gleitzahlen, also position_dbl
2. Umrechnung der Winkelwerte ins Bogenmaß (rad)[49]
3. Anwenden der Formel zur Entfernungsberechnung.

Analog dazu werden die folgenden Rechenoperationen angewandt:

1. $Grad + \frac{Minuten}{60'} + \frac{Sekunden}{3600''}$, liefert die Gleitzahl für einen Winkel in dms_s
2. $\frac{Winkel}{180} * \pi$, liefert den Winkelwert im Bogenmaß (rad)
3. $e = \arccos(\sin\varphi_1 * \sin\varphi_2 + \cos\varphi_1 * \cos\varphi_2 * \cos(\lambda_2 - \lambda_1))$ [50]

   $Entfernung = e * r$, wobei r der Erdradius[51] ist

Hierzu wird in einem Beispiel, im Anhang D, die Entfernung zwischen Frankfurt und Berlin berechnet.

---

[49] Einheit eines in Bogenmaß gemessenen Winkels, Zeichen: rad

[50] Die Formel basiert auf Berechnungen der Einheitskugel. Eine Herleitung ist zu finden auf [W18]. Die griechischen Zeichen λ (Lambda) und φ (Phi) werden in Formeln häufig für die Winkelwerte der Längen- und Breitengrade verwendet.

[51] Je nach Ort und Methode variiert der Erdradius. Hier wird der Wert für den mittleren Erdradius verwendet: 6371,221 km

3.3.6 Erkennen von Sehenswürdigkeiten

Implementierung in der Klasse „JourneyInfo“

Die in Kapitel 3.3.5 gewonnenen Erkenntnisse bezüglich Datentypen und Entfernungsberechnung sind Grundlage für Methoden, zum Erkennen von Sehenswürdigkeiten. Im Folgenden sind zwei solcher Methoden aufgezeigt, die auch in dieser Arbeit verwendet werden. Weitere, ausgedehnte Methoden sind nicht auszuschließen. Welche Methode zum Einsatz kommt kann in Abhängigkeit der Kategorie einer Sehenswürdigkeit entschieden werden.

3.3.6.1 Direkter Umkreis mit Bewegungsrichtung

Beim direkten Umkreis werden aktuelle Positionsdaten[52] des Fahrzeugs mit Positionsdaten aller POI’s auf dem System verglichen. Hierbei kann eine Skalierung für alle drei Vergleichswerte vorgenommen werden. Grundeinstellung ist 5.0° Abweichung für die Bewegungsrichtung und 0.0009° (das sind 3,42 Winkelsekunden) Abweichung für Latitude und Longitude. Das entspricht in etwa 100 Metern, am Äquator[53].

Somit kann das Vorhandensein einer Sehenswürdigkeit, bis auf 100 Meter genau, im Umkreis festgestellt werden. Es wird allerdings kein tatsächlicher Abstand zur Sehenswürdigkeit berechnet, sondern nur ein Umkreis um das Fahrzeug herum. Dieser kann über die Skalierung verändert werden.
Das Einbeziehen der Bewegungsrichtung legt zusätzlich noch fest, dass sich das Fahrzeug in einem bestimmten Winkel zur Sehenswürdigkeit befinden muß. So kann zum Beispiel bei Hinweisschildern unterschieden werden, ob sie in Fahrtrichtung lesbar sind, oder nicht.

Sinnvoll erscheint die Methode des direkten Umkreises, wenn es notwendig ist beim direkten Passieren einer Sehenswürdigkeit zu reagieren. Im folgenden Beispiel ist dargestellt wie der Reiseführer ein Touristisches Hinweisschild an einer Bundesautobahn erkennt.

Die in Abbildung 3.22 dargestellte Landkarte zeigt ein Autobahnteilstück der A45. An der markierten Stelle befindet sich ein Touristisches Hinweisschild, das nur in Fahrtrichtung Dortmund lesbar ist, nicht aber in Richtung Frankfurt. Beim Vorbeifahren, in Richtung Dortmund, soll dem Benutzer zumindest der Text des Schildes vorgelesen werden. Weitere Ausgaben wären optional.

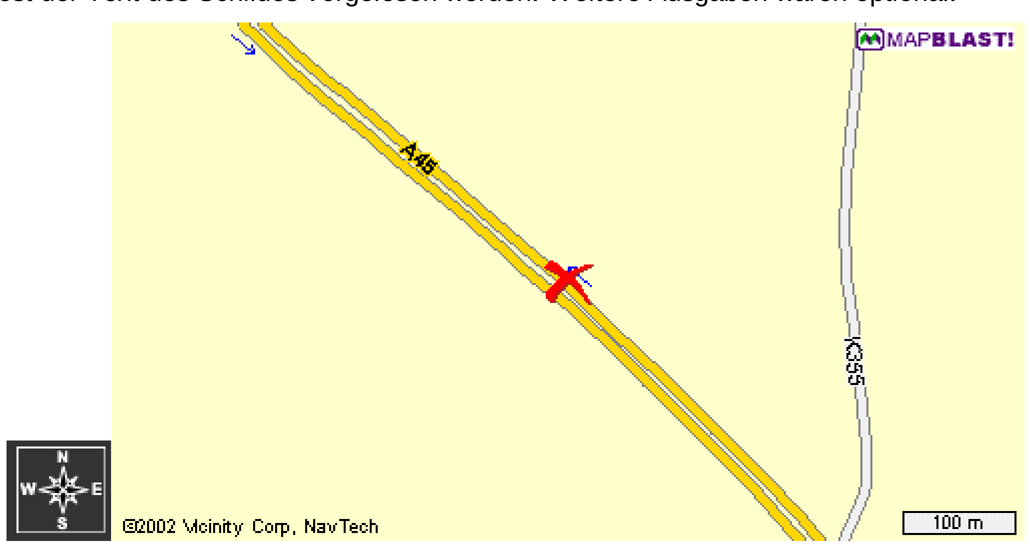


Abbildung 3.22: Touristisches Hinweisschild A45 (100m)

Das selbe Autobahnteilstück wird in Abbildung 3.23, in einem anderen Maßstab, gezeigt. Hier jedoch mit zusätzlichen Markierungen des Fahrzeuges, seinem Umkreis und der Bewegungsrichtung.

---

[52] Latitude, Longitude und Bewegungsrichtung im Format position_dbl

[53] Siehe auch Kapitel 2.3.3 zur Abweichung bei der Positionsbestimmung.

Mit den Koordinaten 8° 34‘ 28“ Ost ($8{,}57\overline{4}^\circ$ O) und 50° 32‘ 38“ ($50{,}543\overline{8}^\circ$ N) ist die Position des Schildes eindeutig festgelegt. Um das Schild, im Vorbeifahren, lesen zu können wird eine Bewegungsrichtung von ca. 312,0° benötigt.

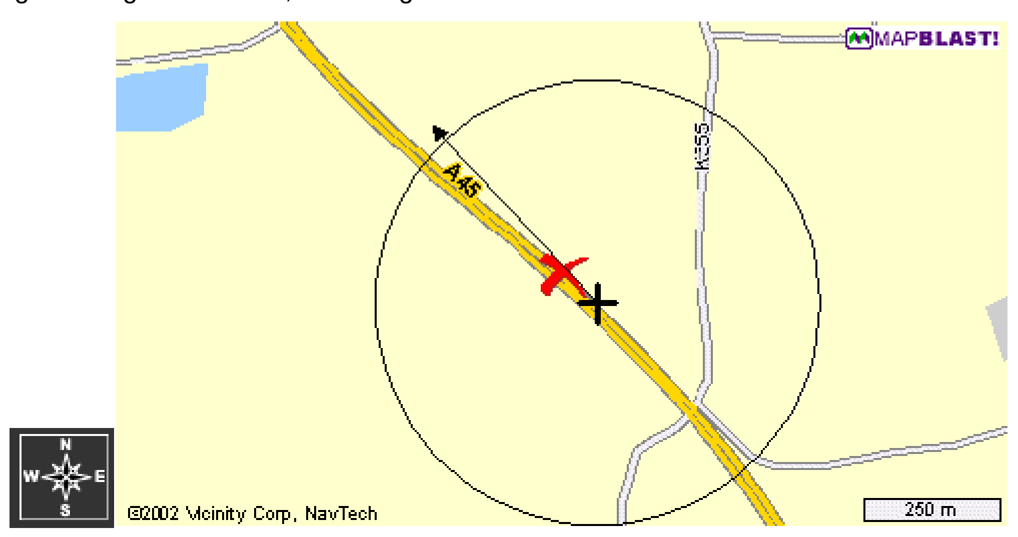


Abbildung 3.23: Touristisches Hinweisschild A45 (250m)

Stimmen Fahrzeugposition und Bewegungsrichtung mit den Werten des Hinweisschildes, bis auf eine geringe Abweichung, überein so wird das Schild als erkannt bewertet. Nun kann die Sprachausgabe des Schildtextes veranlasst werden.

Hier wird deutlich, warum die Bewegungsrichtung eine so große Rolle spielt. Würde sich das Fahrzeug auf der anderen Fahrspur, Richtung Frankfurt befinden, so befände sich das Hinweisschild genauso im Umkreis des Fahrzeugs. Die Bewegungsrichtung wäre aber eine völlig andere, meist genau die entgegengesetzte. Es käme also, völlig zu recht, nicht zur Sprachausgabe. Denn schließlich soll der Hinweistext nur ausgegeben werden, wenn das Schild tatsächlich lesbar ist.

Allerdings ist die Bewegungsrichtung ein zusätzliches Merkmal einer Sehenswürdigkeit, das häufig nicht zur Verfügung steht. Oft enthalten vorhandene Datenbanken von POI’s nur die Koordinaten. Soll die Methode des direkten Umkreises auf ein POI oder eine Kategorie von POI’s angewendet werden, so muß sichergestellt sein, dass die Zusatzinformation vorhanden ist.

3.3.6.2 Annäherung an ein Objekt

Um die Annäherung an ein Objekt (Sehenswürdigkeit) festzustellen, wird tatsächlich eine Entfernung zwischen diesem und dem Fahrzeug berechnet. Unterschreitet der Abstand zu einem Objekt einen bestimmten Wert, z.B. 500 Meter so befindet sich das Objekt in einem definierten Umkreis zum Fahrzeug. Dies ist so ähnlich, wie in der Methode oben, es werden jedoch keine Positionen verglichen, sondern Abstände tatsächlich berechnet. Der erkannte Punkt wird in einer Liste, zusammen mit dem Abstand, vermerkt.

Im weiteren Verlauf wird die Veränderung des Abstandes beobachtet. Zwei Werte, einer für den alten und einer für den neuen Abstand werden vorgehalten. Als absolute Bedingung wird hier vereinbart, das es nur zum Erkennen des Objektes kommt, wenn sich der neue Abstand gegenüber dem alten verringert hat. Zusätzlich führt man noch ein das der neue Abstand wieder einen bestimmten Wert, z.B. 200 Meter unterschreiten muß, damit es tatsächlich zur Erkennung kommt. Auch hier sollte ein kurzes Beispiel den Sachverhalt deutlicher machen.

Im Innenstadtbereich von Wetzlar gibt es eine gehäufte Ansammlung von starren Blitzanlagen[54]. Um dem Reisenden die Blitzanlage rechtzeitig ansagen zu können, muß eine Vorausberechnung statt finden. In Abbildung 3.24 ist eine Karte der Innenstadt, mit den Blitzampeln A, B und C dargestellt.

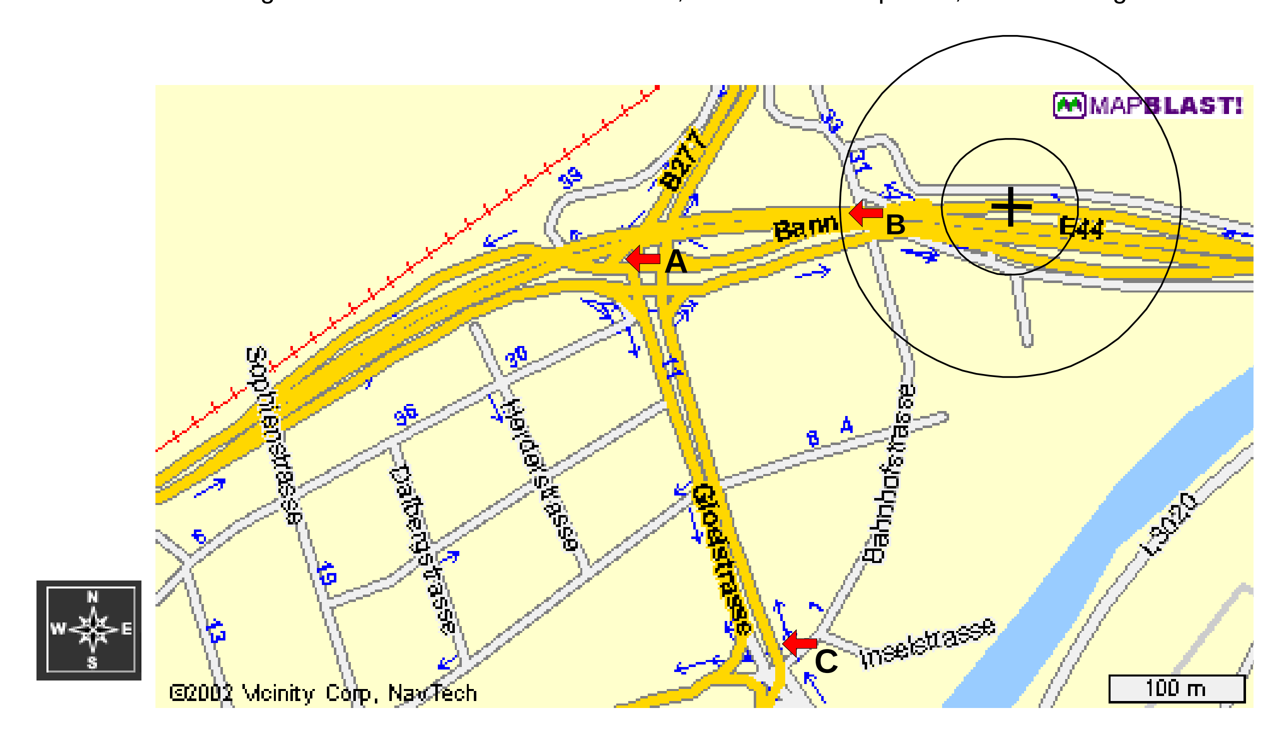


Abbildung 3.24: Blitzanlagen im Innenstadtbereich Wetzlar (100m)

Das von zwei Kreisen umgebene Kreuz stellt die momentane Fahrzeugposition dar. Die Blitzampel B befindet sich bereits innerhalb des äußeren Kreises, also wird der Abstand zu dieser überwacht. Verringert sich im weiteren Verlauf der Abstand zu B und liegt B innerhalb des kleineren Kreises, so wird die Blitzanlage als erkannt vermerkt. Mit der Ausgabe eines Warntons, oder Sprachausgabe kann der Reisende auf die Blitzanlage aufmerksam gemacht werden.

Aus verschiedenen Gründen ist es nicht möglich, die Methode des direkten Umkreises auf das angegebene Beispiel anzuwenden. Ein Warnhinweis bezüglich einer Blitzampel ist dem Benutzer wenig hilfreich, wenn er gerade an ihr vorbei fährt. Demnach ist eine Vorausschau zwingend notwendig.

Man könnte vom Datenlieferant fordern, dass er Punkte angibt die, z.B. 200 Meter, vor einer Blitzampel liegen. Dies scheint aber wenig praktikabel, da Blitzanlagen im Kreuzungsbereich häufig in mehrere Richtungen blitzen. Somit müßte man viele Positionsdaten, inklusive Bewegungsrichtung angeben, um vor einer Blitzanlage zu warnen.

---

[54] Es handelt sich hier um starre Blitzanlagen zur Geschwindigkeitsmessung, oder um sogenannte Blitzampeln. Das Auswerten von Strahlen einer Radarfalle ist strafbar und wird hier weder ermutigt, noch beschrieben. Lediglich die Auswertung von Koordinaten der Blitzanlagen ist Gegenstand dieses Beispiels.

Oder man gelangt zu der Idee nur einen Umkreis zu bestimmen und auf die Bewegungsrichtung zu verzichten. Dies scheitert auch, wenn der Umkreis so groß gewählt wurde, dass sich mehr als eine Blitzanlage in diesem befindet. Siehe hierzu Abbildung 3.25, die einen Auszug aus der Konstellation der oben dargestellten Blitzanlagen wieder gibt.

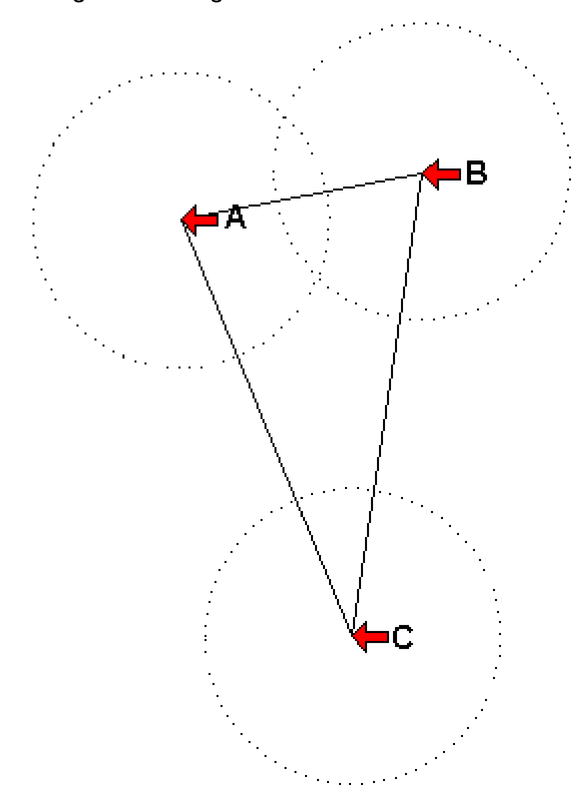


Abbildung 3.25: Blitzanlagen Wetzlar mit Umkreis

Wie eingangs erwähnt sind sicher mehr als die zwei beschriebenen Methoden denkbar. Es erscheint aber nicht als zweckmäßig möglichst viele, oder alle Methoden aufzuführen. Wichtiger ist die Erfahrung, dass das Erkennen von Sehenswürdigkeiten allein durch bestimmen des Umkreises nur zu mäßigem Erfolg führt. Um eine gute Methode zu Implementieren wird es stets Notwendig sein noch weitere Kriterien, wie Bewegungsrichtung (Kapitel 3.3.6.1) oder Distanzverringerung (Kapitel 3.3.6.2), mit einzubeziehen.

3.3.7 Datenbankdateien

Implementierung in den Klassen „Product“ und „POI“

Die sogenannten Datenbankdateien sind als konkrete Lösung bezüglich des Entwurfes aus Kapitel 3.2.6 anzusehen. Es wird zwar kein relationales Datenbanksystem eingesetzt, aber die Tabelleneinträge behalten im Prinzip ihre Gültigkeit. Sie führen zu Textdateien, die mit einfachen Öffne- und Lesebefehlen handhabbar sind. Mittels einer Parsefunktion[55] können Dateieinträge gezielt ausgewählt werden.

3.3.7.1 Die Funktion ParseFile()

Der Funktion ParseFile() wird der Zeiger auf einen Dateinamen übergeben und der Zeiger auf einen Suchtext. Innerhalb der angegebenen Datei sucht die Funktion nach dem Vorkommen des Suchtextes. Die Zeile, in der sich der Suchtext befindet, wird ab dem Suchtext zurückgegeben. Ist die Zeile nach dem Suchtext leer, so wird ein leerer Text zurück gegeben. Konnte der Suchtext in der gesamten Datei nicht gefunden werden, so wird der Suchtext selber zurückgegeben.

[55] Quellcode wurde freundlicherweise von der KWest GmbH zur Verfügung gestellt.

Ein kurzes Beispiel dazu:

- ParseFile::**ParseFile**(const char* filename, char *searchtxt)
- char *searchtxt wird mit **HALLO** geladen
- const char* filename mit **file_hallo.txt**
- Ihalt der Datei file_hallo.txt ist:
- **HALLO Hallo Welt! Das ist die Parsefunktion.**
- Die Parsefunktion findet den Suchtext „HALLO“
- Rückgabe in searchtxt ist „Hallo Welt! Das ist die Parsefunktion.“

Tabelle 3.25: Beispiel Parsefunktion

3.3.7.2 Product-Datei

Durch die Product-Datei werden gleichzeitig die Tabellen Product und POI realisiert. Da jeder POI einem Produkt zugeordnet sein muß wird er direkt in der selben Datei abgelegt.

Eine Product-Datei liegt, auf dem System, in einem bestimmten Verzeichnis und endet immer auf „.prod“. Jede hat einen Dateikopf, der aus Informationen über das Produkt selber besteht. Attribute und deren Bedeutung sind in Tabelle 3.26 angegeben. Ein konkretes Beispiel ist in Tabelle 3.27 zu finden.

Attribut	Bedeutung
PRODUCT_ID	Ganzzahliger Zweibyte Wert, der Produkte eindeutig kennzeichnen soll.
PRODUCT_NAME	Kommerzielle Bezeichnung des Produktes, die möglichst eindeutig zu halten ist.
DESCRIPTION	Name einer HTML-Datei die weitere Beschreibungen über das Produkt enthält.
PRODUCT_CATEGORY	Kennung der Produktkategorie, passend zu einem von: TOUR, COUNTRY, STATE, REGION, CITY, INFO
LANGUAGE	Geeigneter Sprachraum des Produktes. Ein Buchstabe von: D, DK, GB, F, I, NL, E

Tabelle 3.26: Kopf der Product-Datei

PRODUCT_ID **5**
PRODUCT_NAME **Stadtfuehrer Wetzlar**
DESCRIPTION **Stadtfuehrer_Wetzlar.html**
PRODUCT_CATEGORY **CITY**
LANGUAGE **D**

Tabelle 3.27: Beispiel zur Product-Datei Stadtfuehrer_Wetzlar.prod

Es folgen beliebig viele Einträge für POI's. Sie haben die in Tabelle 3.28 beschriebene Form, mit dem Beispiel von Tabelle 3.29. Zwischen den POI Einträgen ist jeweils eine leere Zeile eingefügt.

Attribut	Bedeutung
POI_REF	Ganzzahliger Vierbyte Wert, der POI's eindeutig kennzeichnen soll. Wird gebildet aus der PRODUCT_ID, an die eine fortlaufende Nummer angehängt wird. Diese fortlaufende Nummer beginnt in jeder Product-Datei bei eins.
POI_NAME	Name des POI's, der optional angegeben werden kann.
TOUR_NUMBER	Falls dieser POI Bestandteil einer geführten Tour ist, so ist dies seine Nummer zum Kennzeichnen der Reihenfolge.
POSITION_DMS	Geographische Längen- und Breitenangabe des POI's. Ähnlich der Struktur dms_s.
DIRECTION	Winkel der Bewegungsrichtung, die durchfahren werden sollte, um den POI zur Anzeige zu bringen.
POI_CATEGORY	Kennung der POI-Kategorie. Eins von: RADAR_TRAP, RADAR_TRAFFICLIGHT, SIGN, SIGHTSEEING
ADDRESS	Ortsangaben zum POI in Klartext.
DESCRIPTION	Kurze Beschreibung des POI's.
INFOTEXTS	Strukturierte Textdatei „.info“, die weitere Informationen zum POI enthält.
PICTURES	Pfadangabe, wo evtl. Bilder zum POI abgelegt sind.
JINGLES	Art des Kurztons, der vor der Sprachausgabe gespielt werden soll. Einer von: NO_JINGLE, INFOTEXT, ADVICE, WARNING

Tabelle 3.28: Attribute eines POI's

Da mehr als ein POI in einer Product-Datei eingetragen sein kann werden alle POI relevanten Zeilenmarkierer mit der POI_REF ergänzt. Kommaseparierte Sätze werden auch mit einem Komma abgeschlossen.

POI_REF51 **51**

POI_NAME51 **Dom Wetzlar**

TOUR_NUMBER51

POSITION_DMS51 **008,30,10,04,E,50,33,21,06,N,**

DIRECTION51 **235.0**

POI_CATEGORY51 **SIGHTSEEING**

ADDRESS51 **Deutschland, Hessen, Lahn-Dill-Kreis, 35578, Wetzlar, Domplatz,**

DESCRIPTION51 **Der Dom zu Wetzlar**

INFOTEXTS51 **Dom_Wetzlar.info**

PICTURES51

JINGLES51 **INFOTEXT**

Tabelle 3.29 Beispiel POI Eintrag in Stadtfuehrer_Wetzlar.prod

3.3.7.3 Description-Datei

Die Description-Datei enthält Beschreibungen des Produktes, zu dem sie gehört. Sie endet auf .html und ist nach dem bekannten HTML-Standard[56] aufzubauen. Über die MMI kann ein Benutzer eine solche Datei, mittels Browser, zur Anzeige bringen. Bei dem Browser handelt es sich um eine Ableitung des bekannten „Konqueror" Browsers. Er ist als lauffähiges Programm, auf dem MC5400, verfügbar und wird in dieser Arbeit nicht weiter beschrieben. In nachfolgender Abbildung wird die Description-Datei zum Produkt „Stadtführer Wetzlar" angezeigt.

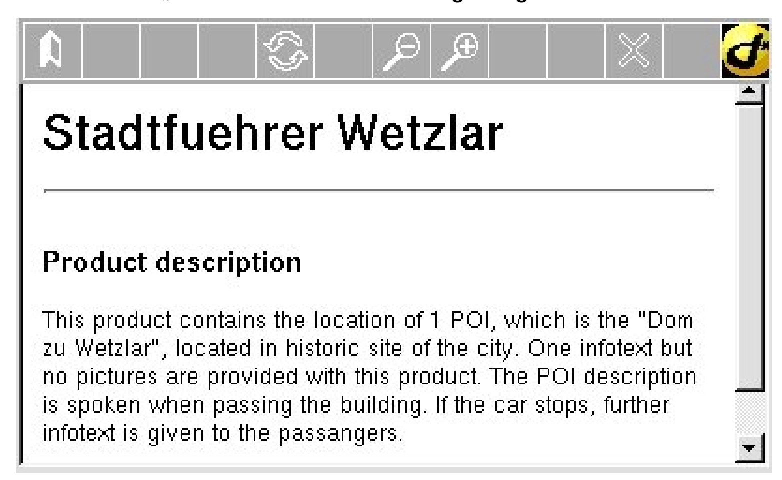


Abbildung 3.26: Beispiel Description-Datei zum „Stadtführer Wetzlar"

3.3.7.4 Info-Datei

Soll zu einer Sehenswürdigkeit ein längerer Text vorgelesen werden, so ist dieser in einer speziellen Textdatei „.info" abgelegt. Auf die Datei wird immer von dem POI aus der Product-Datei verwiesen. Sie enthält die entsprechende POI_REF und eine beliebige Anzahl von beschreibenden Sätzen. Diese Sätze sind gekennzeichnet und fortlaufend numeriert.

Aus verschiedenen Gründen ist es nicht von Vorteil einen langen Text komplett zu synthetisieren und zur Ausgabe zu bringen.

Um zu vermeiden, dass das Computersystem überlastet wird, wird immer nur ein Satz an das Sprachsynthese-System übergeben. Somit werden die Sätze nacheinander synthetisiert und ausgegeben.

Ein Beispiel, in Tabelle 3.30, soll dies verdeutlichen:

POI_REF51 **51**
SENT1 **Weithin sichtbares Wahrzeichen Wetzlars ist der Dom.**
SENT2 **Die heute uebliche Bezeichnung tritt schon im 17. Jahrhundert auf, hat sich aber wohl erst im 18. Jahrhundert eingebuergert.**
SENT3 **Das monumentale, gleichwohl bisher unvollendete Bauwerk bildet den architektonischen Mittelpunkt der Altstadt.**

Tabelle 3.30: Beispiel von Dom_Wetzlar.info

[56] Den Standard definiert das World Wide Web Consortium W3C [W20].

3.3.8 MMI des Reiseführers

Implementierung in der Klasse „GoTo“

Die in Kapitel 3.2.3 gezeigten Screen Prototypes führen zu einer tatsächlichen Implementierung der MMI. Realisiert wird die MMI in der Klasse „GoTo“, die auf der Grafikbibliothek Qt und VDO eigenen Adaptionen basiert. Funktionen und Elemente werden hier im wesentlichen übernommen. Da die Beschreibung der Screen Prototypes sehr ausführlich ist, wird nicht weiter auf das Erscheinungsbild der MMI eingegangen.

Neben der graphischen Benutzerinteraktion werden durch die Klasse GoTo aber noch weitere Aufgaben erledigt. Initialisierungsvorgänge, wie zum Beispiel starten des Sprachservers, werden hier zentral ausgeführt. Starten sämtlicher Leistungsmerkmale wie Reiseinfo, Tourenführung, Browser und Datenbank geschieht auch über Funktionen von GoTo.

Die Einbindung des virtuellen Reiseführers, in das Gesamtsystem des MC5400, ist über eine Konvention möglich. In einem verabredeten Verzeichnis wird ein Hintergrundbild, ein Icon und ein lauffähiges Programm hinterlegt. Hintergrundbild und Icon sind frei wählbar, müssen aber schwarz/weiß mit einem Bit codiert sein. Das lauffähige Programm ist die Gesamtheit aller kompilierten Sourcen des virtuellen Reiseführers, inklusive einer Hauptroutine – main(). Diese wird beim Start des Reiseführers aufgerufen. Nach verschiedenen Initialisierungsschritten kommt es zur Darstellung der MMI. Es wird ein Objekt der oben erwähnten Klasse GoTo erzeugt und zur Anzeige gebracht. GoTo bildet also einen Eintrittspunkt, wie in Abbildung 3.27 dargestellt.

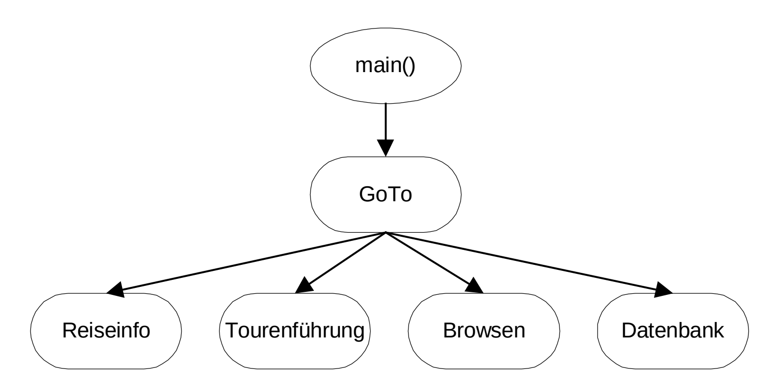


Abbildung 3.27: GoTo als Eintrittspunkt zu den Leistungsmerkmalen

3.3.9 Verbale Reiseinfo

Implementierung in der Klasse „JourneyInfo“

Mit der verbalen Reiseinfo wird eines der möglichen Leistungsmerkmale, aus Kapitel 3.2.1 konkret realisiert. Die Implementierung des Leistungsmerkmals, in der Klasse JourneyInfo, sieht vor, dass eine Verlaufsfunktion existiert, die in eine Dauerschleife übergeht. Diese Funktion wird in einem neu erzeugten Prozess gestartet. Innerhalb der Verlaufsfunktion werden verschiedene Schritte abgearbeitet, die im folgenden erklärt sind. Zum Abbruch der Funktion und beenden des Prozesses gibt es verschiedene Möglichkeiten.

Die Klasse selbst verfügt über eingebettete Objekte der Klassen Geo, GPS, und Speech. Objekte der Klassen Nav und Garmin werden lokal erzeugt, falls notwendig. Somit steht eine Vielzahl der vorher beschriebenen Funktionalitäten zur Verfügung.

3.3.9.1 Schritte der Verlaufsfunktion

Zu Beginn der Verlaufsfunktion wird das GPS Objekt dazu aufgefordert eine geeignete GPS-Quelle auszuwählen. Die Art der gewählten GPS-Quelle wird über einen numerierten Datentyp zurückgegeben, der entweder NONE (keine GPS-Quelle gefunden), NAV_GPS (Navigationssystem) oder GARMIN_MOUSE (Garmin Mouse) sein kann. Über eine Auswahlanweisung wird der Kontrollfluß, abhängig vom GPS Typ, verzweigt.

Konnte keine GPS-Quelle gefunden werden, so wird das GPS Objekt aufgefordert geöffnete Schnittstellen wieder zu schließen. Die Verlaufsfunktion gibt die Kontrolle an die aufrufende Instanz zurück und der Prozess wird beendet.

Im Fall von GARMIN_MOUSE und NAV_GPS ist der Ablauf vom Prinzip her ähnlich, deswegen wird stellvertretend nur der Fall NAV_GPS beschrieben. Ein lokal erzeugtes Objekt der Klasse Nav wird mit dem Filedeskriptor belegt, den das GPS Objekt zur Verfügung stellt. Konnten weitere Initialisierungs- und Konfigurationsschritte erfolgreich durchgeführt werden, so wird in die Dauerschleife verzweigt. Ansonsten wird die Kontrolle zurückgegeben und der Prozess beendet.

Innerhalb der Dauerschleife werden die folgenden Schritte, dem Prinzip nach endlos, wiederholt:

- Aktuelle Fahrzeugposition abfragen
- Längen- und Breitengrad der Fahrzeugposition in Gleitzahlen konvertieren
- Erkennung von Sehenswürdigkeiten durchführen
- Eine Sekunde verweilen, bevor zum Kopf der Schleife zurück gekehrt wird

Zum Erkennen von Sehenswürdigkeiten können verschiedene Methoden, wie in Kapitel 3.3.6 beschrieben, verwendet werden.

3.3.9.2 Abbruchkriterien

Verschiedene Kriterien zum Abbruch der Verlaufsfunktion sind bereits genannt worden. Das Beenden der Verlaufsfunktion verläuft immer nach dem selben Schema. Das Funktionsende wird erreicht, offene Schnittstellen werden geschlossen und der Prozess wird beendet. Aus einem der folgenden Gründe kann das Funktionsende erreicht werden.

- Keine GPS-Quelle gefunden
- Fehlfunktion einer GPS-Quelle zur Laufzeit
- Abbruch durch den Benutzer. Umschaltung von aktiv nach inaktiv

Während Implementierung und Test konnten allerdings noch die folgenden, Systembedingten, Abbrüche festgestellt werden.

- Prozess wurde vorzeitig beendet
- Platz im Hauptspeicher nicht ausreichend

Diese gehören jedoch nicht zu den regulären Abbruchkriterien und können nur durch ein verbessertes Gesamtsystem vermieden werden. Schutzmaßnahmen, z.B. gegen vorzeitiges Beenden des Prozesses, sind nicht verfügbar.

3.3.10 Tourenführung

Die Tourenführung, wiederum eine Leistungsmerkmal aus Kapitel 3.2.1, benutzt die verbale Reiseinfo. Das bedeutet, sobald eine Tour ausgewählt und gestartet wurde, ist die verbale Reiseinfo aktiv. Somit ist sichergestellt, dass alle auf der Tour liegenden Sehenswürdigkeiten erkannt und ausgegeben werden.

Als Kern der Tourenführung bleibt noch die Synchronisation der Wegpunkte, mit dem Navigationssystem. Dafür gibt es eine endständige Funktion die auf eine Anzahl von POI's angewendet wird. Daten der POI's stammen aus der entsprechenden Product-Datei, die durch den Benutzer implizit ausgewählt wurde.

Jeder POI besitzt eine TOUR_NUMBER, die bei Product-Dateien des Typs TOUR, größer/gleich eins und verschieden voneinander sind. Von der kleinsten bis hin zur größten TOUR_NUMBER werden die POI's der Reihe nach, mit Ausnahmen, als Ziele im Navigationssytem gesetzt.

Ausnahmen der Reihenfolge sind:

- Überspringen eines POI's
- Anfahren des vorherigen POI's
- Springen zum ersten oder letzten POI
- Abbrechen der Tourenführung

Ein POI wird als Ziel gesetzt, indem seine Positionsdaten an das angeschlossene Navigationssytem übertragen werden. Die Zielführung startet dann automatisch. Nähert, oder erreicht das Fahrzeug den POI, so erfolgt automatisch eine Ausgabe durch die verbale Reiseführung.

Zum Programmfortschritt (nächsten POI anfahren) ist es allerdings notwendig das der aktuelle POI erreicht wird. Erst dann werden die Daten des nächsten POIs übertragen und es wird weiter geführt. Im Prinzip wäre es also einfach beim Navigationssystem die Anfrage zu stellen, ob das aktuelle Ziel erreicht wurde. Ist dies der Fall kann die Tour fortgesetzt werden. Da es aber baulich nicht immer möglich ist ein Fahrzeug auf den genauen Koordinaten eines POIs zu positionieren müssen Ausnahmen bestimmt werden.

Es ist unter den folgenden Bedingungen möglich den Programmfortschritt zu forcieren:

- Der Benutzer wählt eine der oben angegebenen Ausnahmen zum Springen.
- Nach Sprachausgabe des POI verstreicht eine bestimmte Zeitdauer, in der das Ziel nicht erreicht wurde.
- Eine Tour wurde unterbrochen. Der letzte POI wurde vermerkt und es geht weiter, zum nächsten in der Reihenfolge.

Abbildungsverzeichnis

Tabellenverzeichnis

Quellenverzeichnis

Literaturverzeichnis

Bücher

[B1] Booch, Rumbaugh, Jacobson: Das UML-Benutzerhandbuch, 2. Auflage, ISBN 3-8273-1486-0, Addison-Wesley, Bonn 1999.

[B2] Rumbaugh, Blaha u.a.: Object-Oriented Modeling and Design, ISBN 0-13-630054-5, Prentice-Hall International Inc., New Jersey 1991.

[B3] Irland – Republik Irland & Nordirland, 6. Auflage, ISBN 3-87504-192-5, Karl Baedeker Verlag, Ostfildern 1998.

[B4] Irland – Reiseführer, 2. aktualisierte Auflage, ISBN 3-7701-3536-9, DuMont Buchverlag, Köln 1996.

[B5] Bjarne Stroustrup: Die C++ Programmiersprache, 3. Auflage, ISBN 3-8273-1296-5, Addison-Wesley, Bonn 1998.

[B6] Helmut Herold: Linux – Unix Systemprogrammierung, 2. Auflage, ISBN 3-8273-1512-3, Addison-Wesley, Bonn 1999.

[B7] Brechmann, Dzieia u.a.: Elektrotechnik Tabellen Kommunikationselektronik, 2. Auflage, ISBN 3-14-225037-9, Westermann, Braunschweig 1996.

Festschriften und Sammelwerke

[F1] The Festival Speech Synthesis System, System documentation, Edition 1.4. for Festival 1.4.2, by Alan W. Black, Paul Taylor and Richard Caley, 25. Juli 2002

[F2] The IMS German Festival Manual, for Festival version 1.4.1 with IMS_german version 1.2, Gregor Möhler, Antje Schweitzer, Mark Breitenbücher, 17. Juli 2001

[F3] Definition of low level telegram format for CCI, Author: Ralf Löffert, 26.02.1999, **vertraulich** Siemens VDO Automotive AG

[F4] Product Software Requirements Specification Application Protocol CCI for PC5100 pro CCI SW 2.5, Authoren: Joachim Tröll, Eric Guillard, 25.03.2002, **vertraulich** Siemens VDO Automotive AG

Websites-Verzeichnis

[W1] Südwest Rundfunk – Elektronik im Auto, http://www.swr.de/rasthaus/archiv/2002/02/16/beitrag1.html, 09.07.2002

[W2] SAE Ground Vehicle Technical Comitee, http://www.sae.org, 09.07.2002

[W3] Institute of Electrical and Electronics Engineers, http://www.ieee.org, 05.07.2002

[W4] U.S. Coast Guard Navigation Center, http://www.navcen.uscg.gov/gps/default.htm, 10.07.2002

[W5] LEO – Link Everything Online, search term: “Phonem”, http://dict.leo.org, 12.07.2002

[W6] Cambridge Scientific Abstracts, Linguistics and the law, http://www.csa.com/hottopics/linglaw/gloss.html, 12.07.2002

[W7] The Aerospace Corporation, Education, http://www.aero.org/publications/GPSPRIMER/index.html, 16.07.2002

[W8] Verdand der Automobilinsdustrie VDA, http://www.vda.de 23.07.2002

[W9] The Linux Documentation Project, http://www.tldp.org 24.07.2002

[W10] The Xfree86 Project Inc., http://www.xfree.org 24.07.2002

[W11] Trolltech: Creators of Qt, the cross-platform C++ GUI toolkit, http://www.trolltech.com 24.07.2002

[W12] Busybox, http://www.busybox.net 24.07.2002

[W13] World Geodetic System von 1984, http://www.wgs84.com 01.08.2002

[W14] The Festival Speech Synthesis System, http://www.cstr.ed.ac.uk/projects/festival 02.08.2002

[W15] Festival Source Distribution, http://www.cstr.ed.ac.uk/projects/festival/download.html 02.08.2002

[W16] OpenSource version of IMS German Festival, http://www.ims.uni-stuttgart.de/phonetik/synthesis/festival_opensource.html 02.08.2002

[W17] Das MBROLA Prjekt, http://tcts.fpms.ac.be/synthesis 02.08.2002

[W18] Grundlagen Tirgonometrie http://www.dr-lemke.de/Trigonometrie/trigonometrie.html 05.08.2002

[W19] Mapblast! Digitale Karten, http://www.mapblast.com 06.08.2002

[W20] World Wide Web Consortium W3C, zu HTML, http://www.w3.org/MarkUp 06.08.2002

[W21] Scouting Resources, http://www.scoutingresources.org.uk/compass_magvariation.html 07.08.2002

[W22] freshmeat.net: Project details for gpsd, http://freshmeat.net/projects/gpsd/?topic_id=20 04.09.2002

Sonstige Quellen

[S1] Technical Literature for TFT-LCD module, No. LCY – 96065, Datum: 2. Juli 1996

[S2] Sami Lemmetty: Review of Speech Synthesis Technology, Master Thesis, Helsinki University of Technology, Datum: 30. März 1999

[S3] Sebastian Wendt: Evaluation eines Embedded PC-Systems als Mulitmediaplattform im Kraftfahrzeug, Diplomarbeit, Berufsakademie Mannheim, Datum: 22. September 2000

[S4] Garmin GPS 35/36 TracPak, GPS Smart Antenna, Technical Specification, im Januar 1999

[S5] Touristische Hinweisschilder an Bundesautobahnen, im Land Hessen. Herausgeber: Hessisches Landesamt für Strassen- und Verkehrswesen, Dezernat Strassenaustattung und Elektrotechnik. Stand: März 1999

Anhang A – Klassenbeschreibung

Auf ein Listing des gesamten Quellkodes wird in dieser Arbeit bewusst verzichtet. Um dem Leser aber einen Eindruck vom enormen Umfang der Software zu geben sind nachfolgend alle, für den virtuellen Reiseführer relevanten, Klassen aus Kapitel 3.2.5 angegeben. Zu jeder Klasse gibt es eine Beschreibung und die Attribute bzw. Operationen sind aufgelistet, aber nicht näher beschrieben

Klassenname	GoTo
Beschreibung	Die Klasse „GoTo“ übernimmt hauptsächlich Verwaltungsaufgaben und kann als Interface Klasse zur MMI des MC5400 angesehen werden. Im Bezug auf den Reiseführer übernimmt sie Funktionalitäten im Bereich der MMI und der Initialisierung weiterer Komponenten. Leistungsmerkmale wie die „Reiseinfo“, „Reiseführer Browsen“, oder die „Reiseführung“ werden über die Klasse GoTo aufgerufen.
Attribute **private**	[Viehlzahl von Variablen für die Menüdarstellung weggelassen] ▪ pid_t m_pidFestival ▪ pid_t m_pidJourneyInfo ▪ bool m_bFestivalStarted ▪ Qstring m_strSelectedTour
Operationen **public**	[Funktionen für die Menüdarstellung weggelassen] ▪ void start_festival_server(void) ▪ void stop_festival_server(void) ▪ int get_festival_pid(void) ▪ void stop_journey_info(void) ▪ int link_pois_2_journeyinfo(void)

Tabelle A.1: Klasse „GoTo“

Klassenname	Product
Beschreibung	Durch die Klasse „Product“ werden Objekte der realen Welt, von Reiseführern in Buch Form, repräsentiert. Aufgrund der Zusammensetzung aus sogenannten POI's (Points Of Interest) kann im Prinzip jeder beliebige Reiseführer realisiert werden. Hier ist es allerdings nur möglich mit Instanzen eines vorher bestimmten Produkts zu arbeiten. Neue Produkte können im System nicht erstellt werden, sie werden aus der Datenbank gelesen.
Attribute **private**	▪ int m_nProductID ▪ char *m_strProductName ▪ char *m_strDescription ▪ product_category_e m_eProductCategory ▪ product_language_e m_eProductLanguage ▪ vector <POI *> m_vpPOI_LIST
Operationen **public**	▪ int get_product_id(void) const ▪ char * get_product_name(void) const ▪ char * get_product_description(void) const ▪ product_category_e get_product_category(void) const ▪ product_language_e get_product_language(void) const ▪ void get_poi_list(vector <POI *>& vpPOI_LIST)
Operationen **private**	▪ void AddPOI(POI *pPOI) ▪ int RemovePOI(POI *pPOI) ▪ vector<POI *> *get_poi_list(void)
Link Management Interface	▪ friend void LinkProduct(Product *pProduct, POI *pPOI) ▪ friend void UnLinkProduct(Product *pProduct) ▪ friend void UnLinkProduct(POI *pPOI)

Tabelle A.2: Klasse „Product“

Klassenname	POI
Beschreibung	Jeder Punkt der als Sehenswürdigkeit, oder anderweitig, für den Reiseführer von Interesse ist wird als POI abgelegt. Ein POI wird nicht durch den Reiseführer erzeugt, sondern von ihm genutzt. Er wird bei Bedarf aus der Datenbank gelesen. Es ist notwendig jeden POI einer Instanz der Klasse Product zuzuordnen. So kann ein POI im System eindeutig referenziert werden.
Attribute private	▪ int m_nPOIID ▪ char *m_strPOIName ▪ int m_nTourNumber ▪ position_dms m_dmsPosition ▪ position_dbl m_dblPosition ▪ poi_category_e m_ePOICategory ▪ address_s m_sAddress ▪ char *m_strDescription ▪ infotext_s m_sInfotext ▪ char *m_strPicturePath ▪ jingle_e m_eJingle ▪ Product *m_pProduct ▪ JourneyInfo *m_pJourneyInfo
Operationen public	▪ int get_poi_id(void) const ▪ char *get_name (void) const ▪ int get_tour_number (void) const ▪ position_dms get_position_dms (void) const ▪ position_dbl get_position_dbl (void) const ▪ poi_category_e get_category (void) const ▪ address_s get_address(void) const ▪ char *get_description(void) const ▪ infotext_s get_infotext(void) const ▪ char* get_picture_path(void) const ▪ jingle_e get_jingle(void) const ▪ Product *get_product(void) const ▪ JourneyInfo *get_journey_info(void) const
Operationen private	▪ void SetProduct(Product * pProduct) ▪ void SetJourneyInfo(JourneyInfo* pJourneyInfo)
Link Management Interface	▪ friend void LinkProduct(Product *pProduct, POI *pPOI) ▪ friend void UnLinkProduct(Product *pProduct) ▪ friend void UnLinkProduct(POI *pPOI) ▪ friend void LinkJourneyInfo(JourneyInfo *pJourneyInfo, POI *pPOI) ▪ friend void UnLinkJourneyInfo(JourneyInfo *pJourneyInfo) ▪ friend void UnLinkJourneyInfo(POI *pPOI) ▪ friend bool IsEqual(const POI *pPOI1, const POI *pPOI2)

Tabelle A.3: Klasse „POI“

Klassenname	JourneyInfo
Beschreibung	Durch die „JourneyInfo“ wird das Leistungsmerkmal „Reiseinfo“ realisiert. Der Weg des Reisenden wird kontinuierlich verfolgt. Die aktuelle Position des Fahrzeugs wird mit allen Positionen von POI’s verglichen, die sich auf dem System befinden. Für den Positionsvergleich können verschiedene Methoden eingesetzt werden. Wird ein Vergleich als erfolgreich bewertet, so gibt die Sprachausgabe den beschreibenden Text des POIs wieder. Weitere Aktionen sind von der Art des POIs und Handlungen des Benutzers abhängig.
Attribute private	▪ vector<POI *> m_vpPOI_LIST ▪ quick_reference_s m_psQuickReference[NOF_POI] ▪ near_poi_s m_psNearPOI[NOF_POI] ▪ static int POI_count ▪ static int near_POI_count ▪ static int last_POI_spoken ▪ Speech m_speechobj ▪ Geo m_geoobj ▪ GPS m_gpsobj
Operationen public	▪ void get_poi_list(vector<POI *>& vpPOI_LIST) ▪ int run(void) ▪ void stop(void) ▪ int init_speechobj(void) ▪ void forward_speech(char *pText) ▪ bool get_speech_server_init(void) ▪ void set_leave_while(bool value) ▪ int speek_poi_description(int nPOI_Ref) ▪ void speek_poi_info(int nPOI_Ref) ▪ int check_POI_RADAR(int nPOI_Ref) ▪ void add_poi(int nPOI_Ref, position_dms dmsPosition) ▪ static int get_poi_count(void)
Operationen private	▪ void AddPOI(POI *pPOI) ▪ int RemovePOI(POI *pPOI) ▪ vector<POI *> *get_poi_list(void)
Link Management Interface	▪ friend void LinkJourneyInfo(JourneyInfo *pJourneyInfo, POI *pPOI) ▪ friend void UnLinkJourneyInfo(JourneyInfo *pJourneyInfo) ▪ friend void UnLinkJourneyInfo(POI *pPOI)
Andere	▪ friend void handle_leave_while(int signum)

Tabelle A.4: Klasse „JourneyInfo“

Klassenname	TourGuide
Beschreibung	Der „TourGuide“ ist eine Spezialisierung von JourneyInfo. Der Weg des Reisenden wird als Tour vorgegeben. Falls sich spezielle Datenbankdateien des Typs „Tour“ auf dem System befinden, wählt der Benutzer zunächst eine dieser Touren aus. Bei angeschlossenem Navigationssystem werden Zielkoordinaten vom System an die Navigation übertragen. Auf der Tour befindliche POI’s werden nacheinander abgefahren. Ausgabe und Behandlung der POI’s erfolgt nach den Methoden von JourneyInfo.
Attribute	▪ [Bisher keine Implementierungsdetails]
Operationen	▪ [Bisher keine Implementierungsdetails]

Tabelle A.5: Klasse „TourGuide“

Klassenname	Speech
Beschreibung	„Speech“ ist die Klasse, die eine Schnittstelle zum eingesetzten Text to Speech System bildet. Allgemeine Initialisierungen der Sprachausgabe und Einstellen der verwendeten Sprache können hier gemacht werden. Ausserdem stehen Funktionen zur Verfügung, um während der Laufzeit Veränderungen am Text to Speech System vorzunehmen. Die Funktion der tatsächlichen Sprachausgabe wird hier gekapselt und auszugebender Text wird an das Text to Speech System weitergegeben.
Attribute private	▪ int m_nPortNumber ▪ struct sockadr_in m_sServer ▪ struct hostent *m_pServerData ▪ int m_nSpeechSock ▪ bool m_bSpeechServerInit
Operationen public	▪ int speech_out_init(void) ▪ void language_init(voicelang_e sVoice) ▪ void speech_out_speek(char *pText) ▪ void speech_out_set(char *pText) ▪ void set_port_number(int nPortNumber) ▪ bool get_speech_server_init(void) ▪ int get_speech_sock(void)

Tabelle A.6: Klasse „Speech“

Klassenname	Geo
Beschreibung	Die Klasse „Geo“ liefert geometrische Funktionen für den virtuellen Reiseführer. Datentypen bezüglich Geokoordinaten, für die Positionsbestimmung, sind hier einheitlich definiert. Funktionen für die Umrechnung von Koordinaten, Abstandsberechnung von Positionen und Positionsvergleiche sind in dieser Klasse zusammengefasst.
Attribute	▪ [Keine]
Operationen	▪ double dms_in_double(dms_s dmsValueIn) ▪ dms_s double_in_dms(double dValueIn) ▪ double double_in_rad(double dValueIn) ▪ position_dbl dms_in_double_pos(position_dms dmsPosIn) ▪ position_dms double_in_dms_pos(position_dbl dblPosIn) ▪ position_dbl double_in_rad_pos(position_dbl dblPosIn) ▪ double distance_to_poi_km(position_dbl dblPosCur, position_dbl dblPosPoi) ▪ double distance_to_poi_m(position_dbl dblPosCur, position_dbl dblPosPoi) ▪ int compare_pos_poi_fix(position_dbl dblPosCur, position_dbl dblPosPoi) ▪ int compare_pos_poi_scale(position_dbl dblPosCur, position_dbl dblPosPoi, scale_factor_s NewScaleFactors) ▪ vector<POI *>& vpPOI_LIST compare_pos_poi_list_fix(position_dbl dblPosCur, vector<POI *>& vpPOI_LIST) ▪ vector<POI *>& vpPOI_LIST compare_pos_poi_list_scale(position_dbl dblPosCur, vector<POI *>& vpPOI_LIST, scale_factor_s NewScaleFactors)

Tabelle A.7: Klasse „Geo“

Klassenname	GPS
Beschreibung	„GPS“ gruppiert die verschiedenen Alternativen von GPS-Quellen zusammen. Es ist möglich den Reiseführer entweder mit einem angeschlossenen Navigationssystem (über CCI) oder einem handelsüblichen GPS Empfänger, wie die GPS Mouse von Garmin, zu betreiben. Das Auffinden einer geeigneten GPS-Quelle ist Hauptaufgabe dieser Klasse. Zu diesem Zweck sind Grundfunktionalitäten (z.B. Initialisierung und Verbindungsaufbau) jeder möglichen GPS-Quelle hier realisiert. Individuelle Implementierungen bezüglich einer GPS-Quelle wird dann in einer entsprechenden Klasse gemacht. Derzeit gibt es die Klassen „Nav“, für das Navigationssystem und „Garmin“, für die GPS Mouse von Garmin.
Attribute private	▪ char *m_pDevice ▪ int m_nFileDescriptor ▪ int m_nPortNumber ▪ struct sockadr_in m_sServer ▪ struct hostent *m_pServerDate ▪ int m_nGpsSock ▪ bool m_bGpsServerInit ▪ gps_type_e m_eGpsSource
Operationen public	▪ int get_file_descriptor(void) ▪ void set_file_descriptor(int fd) ▪ void close_file_descriptor(void) ▪ int init_com_port(void) ▪ int cci_init_device(void) ▪ void set_device(char *pNewDevice) ▪ int start_gpsd(void) ▪ void stop_gpsd(void) ▪ int init_gps(void) ▪ void set_port_number(int nNewPortNumber) ▪ bool get_gps_server_init(void) ▪ int get_gps_sock(void) ▪ int find_gps_source(void) ▪ gps_type_e request_gps_type(void) ▪ unsigned char calc_checksum(unsigned char* ucBuffer, int iBufferLength)

Tabelle A.8: Klasse „GPS“

Klassenname	Nav
Beschreibung	Die Funktionen der Klasse „Nav“ bilden das Interface zum Navigationssystem. Sie basieren auf den Vorgaben des CCI Kommunikationsprotokolls. Es können Konfigurationen an der Art der Datenübertragung vorgenommen werden. Telegramme für die Positionsbestimmung und das setzen neuer Zielkoordinaten werden mit dem Navigationssystem ausgetauscht.
Attribute private	▪ position_telegram_s m_sPositionTelegram ▪ position_dms m_dmsPosition ▪ date_time_s m_currentDateTime ▪ double m_fHeading ▪ short m_nVelocity ▪ gps_status_s m_sGpsStatus ▪ unsigned char m_bIndex ▪ unsigned char m_bInitIndex
Operationen public	▪ int init_telegram_index (void) ▪ int configure_interface (void) ▪ int start_automatic_pos_update (void) ▪ int stop_automatic_pos_update (void) ▪ void receive_position_telegram (void) ▪ void analyse_position_telegram (void) ▪ position_dms get_position (void) ▪ date_time_s get_date_time (void) ▪ double get_heading (void) ▪ short get_velocity (void) ▪ gps_status_s get_gps_status (void) ▪ int send_new_destination (position_dms newDestination)

Tabelle A.9: Klasse „Nav“

Klassenname	Garmin
Beschreibung	Stellt das Interface zu Garmin’s GPS Mouse her, ähnlich wie Nav zum Navigationssystem. Ein Objekt von Garmin liest kontinuierlich die von der GPS Mouse gesendeten Daten. Diese werden analysiert und aufbereitet.
Attribute private	▪ char *m_nmeaData ▪ position_dms m_dmsPosition ▪ date_time_s m_currentDateTime ▪ double m_fHeading ▪ double m_dVelocity ▪ garmin_gps_status_s m_sGpsStatus ▪ int m_nSatView
Operationen public	▪ void receive_nmea_data(void) ▪ void analyse_nmea_data(void) ▪ void convert_RMC(char *strBuffer) ▪ void convert_GSV(char *strBuffer) ▪ position_dms get_position(void) ▪ date_time_s get_date_time(void) ▪ double get_heading(void) ▪ double get_velocity(void) ▪ garmin_gps_status_s get_receiver_status(void) ▪ int get_satelites_in_view(void)

Tabelle A.10: Klasse „Garmin“

Anhang B - CCI Telegramm für Positionsdaten

Feld	Wert/Bereich	Beschreibung
Function Code	0x60H	Funktionscode, der anzeigt das Positionsdaten gesendet werden. Der hier erwartete Wert ist 0x60 Hex.
DB0	0x01H	Position als Koordinaten
DB1	Bit 0 = 0	kein GPS Empfang
	Bit 0 = 1	mindestens 2 verfügbare Satelliten
	Bit 1 = 1	3D Empfang
	Bit 2,3	Reserviert für zukünftige Verwendungen
	Bit 4 = 0	Position ist Karteninterpoliert
	Bit 4 = 1	Position ist nicht Karteninterpoliert, wegen zu geringer Auflösung
	Bit 5 = 1	Navigationssystem kann keine korrekte Position bestimmen
	Bit 6,7 = 0	Reserviert für zukünftige Verwendungen
DB2	0 bis 23	Stundenanteil der aktuellen Uhrzeit.
DB3	0 bis 59	Minutenanteil der aktuellen Uhrzeit.
DB4	1 bis 31	Tag des aktuellen Datums.
DB5	1 bis 12	Monat des aktuellen Datums.
DB6	0 bis 255	Jahr des aktuellen Datums, als Offset zu 1990. Das Jahr 2002 wird hier also durch eine zwölf dargestellt.
DB7	0 bis 90	Winkelangabe für den Breitengrad.
DB8	0 bis 59	Minutenangabe für den Breitengrad.
DB9	0 bis 59	Sekundenangabe für den Längengrad.
DB10	Bit 0 bis 3	Zehntelsekunde für den Breitengrad.
	Bit 7 = 1	Angaben beziehen sich auf nördliche Hemisphäre
	Bit 7 = 0	Angaben beziehen sich auf südliche Hemisphäre
DB11	0 bis 180	Winkelangabe für den Längengrad.
DB12	0 bis 59	Minutenangabe für den Längengrad.
DB13	0 bis 59	Sekundenangabe für den Längengrad.
DB14	Bit 0 bis 3	Zehntelsekunde für den Längengrad.
	Bit 7= 1	Angaben beziehen sich auf östliche Hemisphäre
	Bit 7= 0	Angaben beziehen sich auf westliche Hemisphäre
DB15 DB16	0 bis 3599	Gemeinsame Nutzung für die Fahrtrichtung, in Zehntelgrad.
DB17	0 bis 255	Geschwindigkeit in km/h.

Tabelle B.1: Datenfelder des Telegramms für Positionsdaten

Anhang C – Installation von Festival

Benötigte Dateien

Von der Internet-Ressource [W15] sind mindestens folgende Dateien herunterzuladen:

- festival-1.4.2-release.tar.gz, Festival Speech Synthesis System Quellcode
- speech_tools-1.2.2-release.tar.gz, Die Edinburgh Speech Tools Bibliothek
- festlex_NAME.tar.gz, Die Lexicon Distribution
- festvox_NAME.tar.gz, Sprachdatenbank

Für Deutsch müssen ausserdem noch weitere Dateien von [W16] kopiert werden:

- ims_german_1.2-os.tgz, Quellcode und Scheme-Dateien für IMS German Festival
- ims_german_1.2-doc.tgz, Dokumentation für IMS German Festival
- bomp_full.tgz, BOMP Lexicon der Universität Bonn
- festival-1.4.1-fixes.tgz

Zusätzlich wird noch der MBROLA Engine und eine Sprache (Deutsch) des MBROLA Projektes benötigt. Dateien können kopiert werden von [W17]. Für diese Arbeit wurden verwendet:

- mbr301h.zip, MROLA Binary für Linux i386
- de2-990106.zip, eine der drei deutschen Sprachdatenbanken

Schrittweises Vorgehen

Grundsätzlich sind drei bzw. vier Komponenten zu installieren. Zu jeder Komponente finden sich, nach dem Entpacken des jeweiligen Dateiarchivs, Installationshinweise. Diesen sollte zusätzlich große Beachtung zu kommen.

Im folgenden wird mit Stichpunkten die Installation der Komponenten beschrieben. Probleme die mit der Systemkonfiguration des verwendeten Linux-Systems zusammen hängen können sehr individueller Natur sein und werden hier nicht ausführlich behandelt. Bei mit „unix$“ angeführten Zeilen handelt es sich um Eingaben auf der Kommandoebene.

- Speech Tools
  - Sourcen in beliebigem Verzeichnis entpacken

    unix$ gzip –d speech_tools-1.2.2-release.tar.gz

    unix$ tar –xvf speech_tools-1.2.2-release.tar

    unix$ cd speech_tools

    unix$ ./configure

    unix$ make
  - Falls Linker-Fehler bzgl. libtermcap, diese via Yast2 installieren und/oder libtermcap.a, libtermcap.so, libtermcap.so.2, libtermcap.so.2.0.8 nach ../speech_tools/lib kopieren.
  - Pfadvariablen in .bashrc (oder geeignete Stelle) anpassen PATH=~/speech_tools/bin:$PATH

- Festival
  - Sourcen in beliebigem Verzeichnis entpacken

  unix$ gzip –d festival-1.4.2-release.tar.gz

  unix$ tar –xvf festival-1.4.2-release.tar

  unix$ cd festival

  unix$ ./configure

  unix$ make

  - Pfadvariablen in .bashrc (oder geeignete Stelle) anpassen
    PATH=~/festival/bin:$PATH
  - Lexicons im Oberverzeichnis von festival entpacken

  unix$ gzip –d festlex_NAME.tar.gz

  unix$ tar –xvf festlex_NAME.tar

  - Sprachen im Oberverzeichnis von festival entpacken

  unix$ gzip –d festvox_NAME.tar.gz

  unix$ tar –xvf festvox_NAME.tar

- German Festival bzw. MBROLA
  - unzip mbr301h.zip in beliebigem Verzeichnis, empfohlen: /usr/local/MBROLA
  - mbrola-linux-i386 in mbrola umbenennen
  - Sprache unzippen z.B. de2-990106.zip

  - ims_german_1.2c-os.tgz im Basisverzeichnis zu Festival entpacken
  - bomp_full.tgz ebenfalls
  - festival-1.4.1-fixes.tgz auch
  - Füge zu festival/config/config hinzu: ALSO_INCLUDE += ims_german_text
  - Setze Pfadnamen von MBROLA und Progname in Datei festival/lib/sitevars.scm

    (set! mbrola-path “usr/local/MROLA“)

    (set! mbrola_progname (string-append mbrola-path “mbrola –e”))
  - Lade Deutsche Module in festival/lib/siteinit.scm

    (require ‘ims_german_opensource)
  - Recompile speech_tools
  - Recompile festival

Anhang D – Beispiel zur Entfernungsberechnung

Entfernungsberechnug zwischen Frankfurt und Berlin nach der vorher beschriebenen Formel:

Koordinaten für Frankfurt: 50° 06‘ 44,0“ nördlicher Breite (φ_1)

08° 40‘ 55,0“ östlicher Länge (λ_1)

Koordinaten für Berlin: 52° 31‘ 20,0“ nördlicher Breite (φ_2)

13° 17‘ 51,0“ östlicher Länge (λ_2)

1. Umrechnen der Winkelwerte in Gleitzahlen:

φ_1 (Breite Frankfurt): $50° + \frac{6'}{60'} + \frac{44''}{3600''} = 50{,}11\overline{2}°$

λ_1 (Länge Frankfurt): $8° + \frac{40'}{60'} + \frac{55''}{3600''} = 8{,}6819\overline{4}°$

φ_2 (Breite Berlin): $52° + \frac{31'}{60'} + \frac{20''}{3600''} = 52{,}5\overline{2}°$

λ_2 (Länge Berlin): $13° + \frac{17'}{60'} + \frac{51''}{3600''} = 13{,}2975°$

2. Winkel in Bogenmaß umrechnen:

φ_1 in rad: $\frac{\varphi_1}{180} * \pi = \frac{50{,}11\overline{2}°}{180} * \pi \approx 0{,}87462°$

λ_1 in rad: $\frac{\lambda_1}{180} * \pi = \frac{8{,}6819\overline{4}°}{180} * \pi \approx 0{,}15153°$

φ_2 in rad: $\frac{\varphi_2}{180} * \pi = \frac{52{,}5\overline{2}°}{180} * \pi \approx 0{,}91669°$

λ_2 in rad: $\frac{\lambda_2}{180} * \pi = \frac{13{,}2975°}{180} * \pi \approx 0{,}23209°$

3. Ausrechnen der Entfernung:

$$e = \arccos(\sin\varphi_1 * \sin\varphi_2 + \cos\varphi_1 * \cos\varphi_2 * \cos(\lambda_2 - \lambda_1))$$

$$e = \arccos(\sin 0{,}87462° * \sin 0{,}91669° + \cos 0{,}87462° * \cos 0{,}91669° * \cos(0{,}23209° - 0{,}15153°))$$

$$e = 0{,}06559$$

$$Entfernung = 0{,}06559 * 6371{,}221km = 417{,}89km$$

Tiefergehende Informationen zur Sphärischen Trigonometrie und exakte Herleitung der verwendeten Formel, zur Entfernungsberechnung, sind zu finden auf [W18].

Anhang E – Ausgewähltes Beispiel

Bei dem folgenden Beispiel wird mit den „Touristischen Hinweisschildern an Bundesautobahnen“ gearbeitet. Dem Beispiel geht ein realer Versuch voraus, der sich im wesentlichen mit der Sammlung von benötigten Daten beschäftigt. Die gesammelten Daten werden dann im Beispiel verwendet.

Datensammlung

Wenn man sich auf Bundesautobahnen bewegt fallen einem außer den blauen Hinweisschildern für Autobahnkreuze, Entfernungen und Ausfahrten noch andere Schilder auf. Es handelt sich dabei um, braune, Touristische Hinweisschilder. Sie machen aufmerksam auf Sehenswürdigkeiten in Nähe der Autobahn oder einer angrenzenden Stadt. Jede Stadt oder Gemeinde kann eine Genehmigung für die Aufstellung solcher Hinweisschilder selbst beantragen. Hierzu muß ein Antrag beim jeweiligen Landesamt gestellt werden. Eine Kommission für touristische Aspekte entscheidet dann über Form und Layout des Schildes und ordnet dessen Aufstellung an. Von einer lokalen Behörde, zumeist der Straßenverkehrsbehörde, wird dann der Standort geprüft und die Aufstellung an einem günstigen Ort durchgeführt. Hierbei wird der Standort nicht mit Geokoordinaten vermessen, sondern lediglich die Betriebskilometer des Autobahnteilstücks werden notiert. Diese Tatsache macht es notwendig zunächst eine Datensammlung durchzuführen. Normalerweise würde diese Aufgabe einem Datenlieferant zukommen.

Von Interesse sind alle Touristischen Hinweisschilder auf einem Autobahnteilstück der A45 zwischen Wetzlar und Dillenburg. Abbildung E.1 ist ein Auszug aus der Karte [S5] für Touristische Hinweisschilder an Bundesautobahnen, die freundlicherweise vom Hessischen Landesamt für Strassen- und Verkehrswesen zur Verfügung gesellte wurde. Sie dient zur einfachen Orientierung, beim Auffinden der Schilderstandorte.

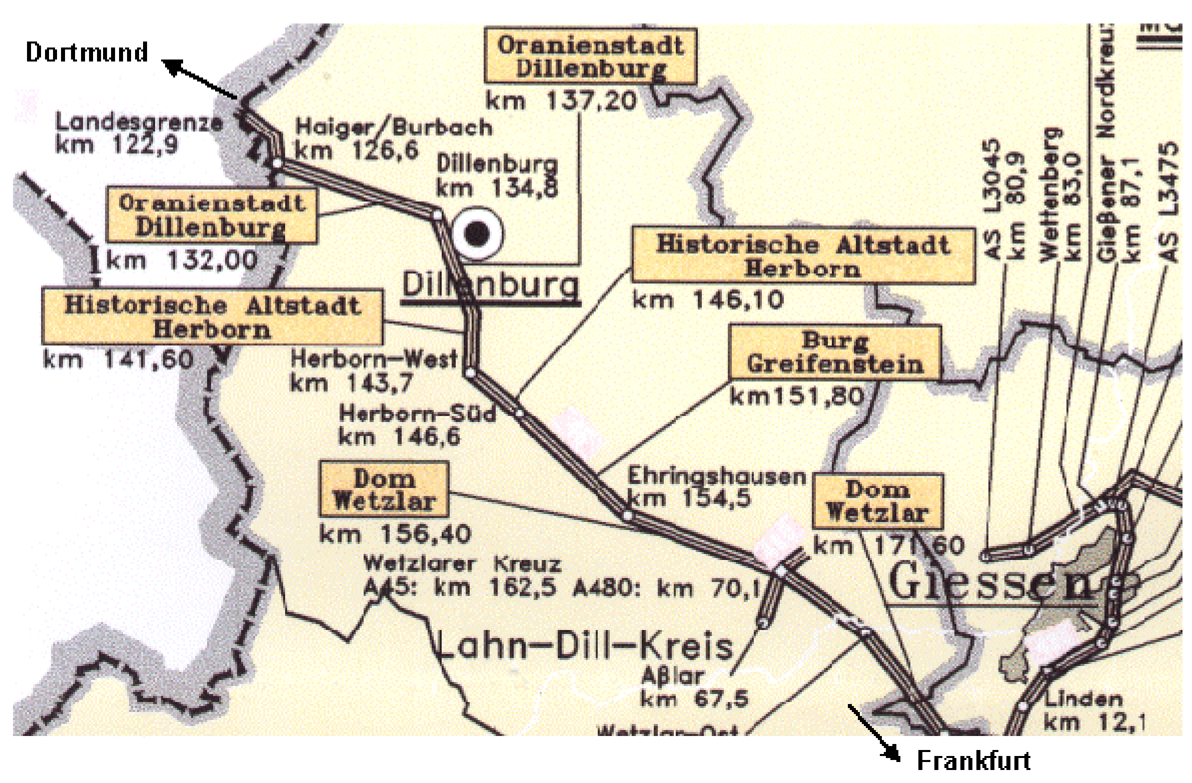


Abbildung E.1: Touristische Hinweisschilder, A45 Wetzlar - Dillenburg

Wie oben erwähnt mußte zunächst eine gründliche Datensammlung durchgeführt werden. Nach mehreren Testfahrten, unter Verwendung eines Navigationssytems, konnten die in Tabelle E.1 dargestellten Ergebnisse ermittelt werden. Hierbei wurde die A45 zwischen Wetzlar und Dillenburg, in Richtung Dortmund bzw. Frankfurt, befahren. Es wurden Positionsdaten und Winkel der Bewegungsrichtung für jedes Hinweisschild aufgenommen.

Professionelle Datensammler würden sich hier allerdings nicht auf die Angaben eines Navigationssystems verlassen, sondern DGPS-Methoden, zur genauen Ortsbestimmung, verwenden. Für den Versuch und das Beispiel sollen die folgenden Daten aber als hinreichend genau betrachtet werden.

#	Name - Hinweisschild	BAB Km	Längengrad	Breitengrad	Fahrtrichtung	Winkel
1	Dom Wetzlar	171,60	8° 34' 28'' Ost	50° 32' 38'' Nord	Dortmund	312°
2	Dom Wetzlar	156,40	8° 24' 35'' Ost	50° 36' 51'' Nord	Frankfurt	122°
3	Burg Greifenstein	151,80	8° 20' 58'' Ost	50° 37' 29'' Nord	Dortmund	278°
4	Historische Altstadt Herborn	146,10	8° 18' 28'' Ost	50° 39' 46'' Nord	Dortmund	279°
5	Historische Altstadt Herborn	141,60	8° 17' 15'' Ost	50° 42' 28'' Nord	Frankfurt	184°
6	Oranienstadt Dillenburg	137,20	8° 16' 42'' Ost	50° 43' 43'' Nord	Dortmund	322°
7	Oranienstadt Dillenburg	132,00	8° 13' 15'' Ost	50° 44' 49'' Nord	Frankfurt	103°

Tabelle E.1: Datensammlung, A45 Wetzlar - Dillenburg

Aus diesen Daten wurde eine komplette Product-Datei erstellt. Der Inhalt der Datei „A45_Schilder.prod“ ist in den folgenden Tabellen E.2 und E.3 dargestellt. Die Datei wird später auf dem MC5400 verfügbar sein. Zunächst der Kopf der Product-Datei:

PRODUCT_ID **1** PRODUCT_NAME **Hinweisschilder A45** DESCRIPTION **A45_Schilder.html** PRODUCT_CATEGORY **INFO** LANGUAGE **D**

Tabelle E.2: Kopf der Datei „A45_Schilder.prod“

Es folgen die Einträge für die Sehenswürdigkeiten, hier sind das die Touristischen Hinweisschilder. Der Übersicht wegen wurde nur ein Eintrag, als Beispiel, ausgewählt. Bedeutung der Inhalte kann in Kapitel 3.3.7, zu den Datenbankdateien, nachgelesen werden.

POI_REF11 **11** POI_NAME11 **Dom Wetzlar** TOUR_NUMBER11 **0** POSITION_DMS11 **008,34,28,00,E,50,32,38,00,N,** DIRECTION11 **312.0** POI_CATEGORY11 **SIGN** ADDRESS11 DESCRIPTION11 **Der Dom zu Wetzlar** INFOTEXTS11 PICTURES11 **A45_Dom.gif** JINGLES11 **ADVICE**

Tabelle E.3: POI Einträge der Datei „A45_Schilder.prod“

Konkretes Beispiel

Ziel des Beispiels ist es nun alle Touristischen Hinweisschilder auf dem in Abbildung E.1 dargestellten Autobahnteilstück zu erkennen und einen kurzen, beschreibenden Text vorzulesen. Dabei ist es wichtig nur die in Fahrtrichtung lesbaren Schilder zu erkennen und doppelte Nennungen zu vermeiden.

Im folgenden Text werden die, vom Benutzer, auszuführenden Aktionen und entsprechenden Reaktionen des Systems, in Stichpunkten, beschrieben. Auf explizites Kennzeichnen von Abbildungen wird hier bewusst verzichtet.



Ein MC5400 wird wie vorgesehen im Fahrzeug verbaut. Eine Softwareversion mit virtuellem Reiseführer ist geladen und die Datei „A45_Schilder.prod" befindet sich auf dem System. Die Funktion „Reiseinfo" wird per MMI aktiviert. Eine Tour ist zu diesem Zeitpunkt typischerweise nicht ausgewählt, oder gestoppt.



Die A45 wird mit dem vorbereiteten Fahrzeug, z.B. ab dem Giessener Südkreuz, in Richtung Dortmund befahren.



Bei Autobahnkilometer 171,60 wird zum ersten Mal ein Touristisches Hinweisschild passiert. Stimmen Positionsdaten und Bewegungsrichtung, bis auf eine minimale Toleranz, überein so wird der Text

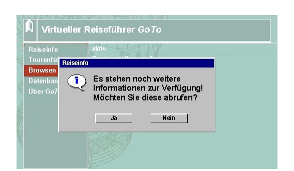
„Der Dom zu Wetzlar" synthetisiert und Ausgegeben.

Da keine weiteren Infotexte zur Verfügung stehen wird keine ausgedehnte Reiseinformation vorgetragen.



Nach etwa 20 Kilometer Fahrt folgt das zweite Schild, bei Autobahnkilometer 151,80. Nach Erkennung, des Schildstandortes, wird hier der knappe Text

„Burg Greifenstein" ausgegeben.



Ausserdem gibt es noch einen Verweis auf einen längeren Infotext, zur Sehenswürdigkeit. In einem Dialogfenster wird der Benutzer gebeten das Vortragen des Infotexts zu bestätigen. Ist dies der Fall wird noch der folgende Text synthetisiert und ausgegeben:

„Die Burg Greifenstein ist das Wahrzeichen der Gemeinde. Schon von weitem laden die imposanten Doppeltürme der Burg zu einem Besuch ein. Als ehemalige Residenz der Grafen zu Solms-Greifenstein und Herrensitz der Nassauer weist das um 1200 urkundlich erstmals erwähnte Bauwerk eine beeindruckende Historie auf."

Historische Altstadt
Herborn



Im weiteren Verlauf, der Autobahn, werde noch zwei Hinweisschilder passiert. Es werden die Texte

„Historische Altstadt Herborn" und

„Oranienstadt Dillenburg" vorgelesen.

Dies geschieht an den betreffenden Stellen, bei BAB Kilometer 146,10 bzw. BAB Kilometer 137,20. Zusätzliche Infotexte stehen bei beiden nicht zur Verfügung.



Bereits während dem Befahren, der A45 in Richtung Dortmund, wurden außer den oben genannten Schildern noch andere passiert. Es handelt sich dabei um die Schilder, die nur in Fahrtrichtung Frankfurt lesbar sind. Der Algorithmus zum erkennen von Sehenswürdigkeiten[57] hat deren Ausgabe folgerichtig verhindert.

Um aber noch zur Ausgabe der Schilder, in Fahrtrichtung Frankfurt, zu kommen wird die Autobahnseite gewechselt.

Oranienstadt Dillenburg

Historische Altstadt Herborn

Dom Wetzlar

Hier kommt es, in Fahrtrichtung Frankfurt, zur Ausgabe der folgenden Texte:

„Oranienstadt Dillenburg“, bei BAB Kilometer 126,6

„Historische Altstadt Herborn“, bei BAB Kilometer 141,60

„Der Dom zu Wetzlar“, bei BAB Kilometer 156,40

Ein Hinweisschild für die Burg Greifenstein existiert in dieser Fahrtrichtung nicht. Demnach kommt es auch zu keiner Ausgabe.

Anmerkung: Alle Hinweisschilder mit der Aufschrift „Dom Wetzlar“ sind mittlerweile gegen „Historische Altstadt Wetzlar“ ausgetauscht worden. Die vom Hessischen Landesamt zur Verfügung gestellte Karte ist auf dem Stand vom März 1999.

---

[57] Hier wurde die Methode „Direkter Umkreis mit Bewegungsrichtung“ aus Kapitel 3.3.6.1 verwendet.